



D7.2.1 SEKT Methodology: Initial Lessons Learned and Tool Design

**Denny Vrandečić (AIFB), York Sure (AIFB),
Christoph Tempich (AIFB), Michael Engler (AIFB, BT)**

with contributions from

**Richard Benjamins (iSOCO), Mercedes Blázquez (iSOCO),
Kalina Bontcheva (U Sheffield), Pompeu Casanovas (UAB), Núria Casellas (UAB),
Miha Grčar (JSI), Heiko Haller (AIFB), Aleks Jakulin (JSI), Nick Kings (BT),
Markus Krötzsch (AIFB), Atanas Kyrkakov (OntoText),
Diana Maynard (U Sheffield), Dunja Mladenić (JSI), Sofia Pinto (U Lisbon),
Marta Poblet (UAB), Stefan Rudlof (AIFB), Valentin Tablan (U Sheffield),
Ian Thurlow (BT), Ralph Traphöner (empolis), Joan-Josep Vallbé (UAB),
Max Völkel (AIFB), Wolf Winkler (AIFB)**

Abstract.

EU-IST Integrated Project (IP) IST-2003-506826 SEKT

Deliverable D7.2.1 (WP7)

From the case studies, we collect the experiences with using the SEKT methodology and the three SEKT technologies SEKT – Human Language Technologies, Ontology Management and Knowledge Discovery. We also describe tool designs and developments based on this experience.

Keyword list: experience, methodology, human language technologies, ontology, knowledge discovery, applied technology, tools

Document Id. SEKT/2005/D7.2.1/v1.0
Project SEKT EU-IST-2003-506826
Date December 27th, 2005
Distribution public

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.

Orion 5/12, Adastral Park
Ipswich IP5 3RE, UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute

Jamova 39
1000 Ljubljana, Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield

Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP, UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.

Pedro de Valdivia, 10
28006 Madrid, Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH

Amalienbadstr. 36
76227 Karlsruhe, Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)

Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Siemens Business Services GmbH & Co. OHG

Otto-Hahn-Ring 6
81739 Munich, Germany
Tel: +49 89 636 40 107, Fax: +49 89 636 40 233
Contact person: Dirk Ramhorst
E-mail: dirk.ramhorst@siemens.com

Empolis GmbH

Europaallee 10
67657 Kaiserslautern, Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB

Englerstr. 28
D-76128 Karlsruhe, Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck

Institute of Computer Science
Technikerstraße 13
6020 Innsbruck, Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH

Tal
6464 Springen, Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Sirma Group Corp., Ontotext Lab

135 Tsarigradsko Shose
Sofia 1784, Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autònoma de Barcelona

Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona, Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Executive Summary

The vision of SEKT is to develop and exploit the knowledge technologies which underlie Next Generation Knowledge Management. We envision knowledge workplaces where the boundaries between document management, content management, and knowledge management are broken down, and where knowledge management is an effortless part of day to day activities. SEKT is built around the synergy of the complementary know-how of the key European centres of excellence in Ontology and Metadata Technology, Knowledge Discovery, and Human Language Technology. The execution of SEKT is based on the integration of fundamental research, component development and integration driven by real world case studies in the public and private sectors. SEKT will provide new insights on knowledge technologies.

The main objective of work package 7 is to provide a methodology for the application of Semantic Knowledge Technologies into enterprises in a goal-driven and process-oriented way. The developed SEKT methodology is applied and evaluated as part of the case studies. In this report we collect the feedback from the case study partners. We present the enhancements of the methodology and include lessons learned and best practices.

Based on this experiences we also designed two DILIGENT-based tools. EvA is meant for the distributed engineering of ontologies, whereas the Semantic MediaWiki is best suited for the population of ontologies. We describe the design of these tools and some application scenarios.

Contents

1	Introduction	7
1.1	Motivation	7
1.2	The SEKT Big Picture	8
1.3	Organization of this deliverable	8
I	Lessons learned	11
2	Knowledge Discovery	13
2.1	The Knowledge Discovery Process: Best Practices	14
2.2	Getting and Preparing the Data	15
2.2.1	Availability of Data	15
2.2.2	Preprocessing the Data in the Legal Case Study	16
2.2.3	Lessons Learned	17
2.3	Using Knowledge Discovery for Ontology Evaluation	18
2.3.1	Evaluation of Ontologies in the Legal Case Study	20
2.3.2	Lessons Learned	24
3	Human Language Technologies	27
3.1	Using HLT in End-User Applications	27
3.2	Automatic annotation	31
3.3	Analysing competency questions	32
3.4	Lessons learned	32
4	Ontology Management Technologies	35
4.1	Basing an Ontology on a Foundational Ontology	35
4.1.1	The Structure of PROTON	35
4.1.2	OPJK Integration	36
4.1.3	Class Integration	36
4.1.4	Inherited Relations	37
4.1.5	Lessons learned	38
4.2	Managing Ontologies	39
4.2.1	Lessons learned	39

5	Methodology	41
5.1	Overview of the SEKT methodology	42
5.2	Social Issues of Ontology Engineering	43
5.2.1	Capturing Professional Judicial Knowledge	44
5.2.2	Lessons learned	45
5.3	Using off the shelf tools	46
5.3.1	Ontology engineering in a wiki	46
5.3.2	Lack of tool integration	47
5.3.3	Ontology maintenance in a forum	48
5.3.4	Lessons learned	48
5.4	Faster Ontology Engineering	49
5.4.1	Focussing the discussion	49
5.4.2	Experiences from ontology building	50
5.4.3	Capturing the argumentation	51
5.4.4	Lessons learned	52
II	Tools	55
6	EvA tool design	57
6.1	Overview of Application's Functionality	58
6.2	Personas	59
6.2.1	Persona Justine	59
6.2.2	Notes on Persona Jack	59
6.2.3	Notes on Persona Richard	59
6.3	Scenarios	60
6.3.1	Learning the application	60
6.3.2	Setting up a new Ontology Development Project for Distributed Partners	61
6.3.3	Arguing over an ontology in an online-setting	62
6.3.4	Arguing over an ontology in a group meeting room with a beamer	64
6.3.5	Creating an ontology with the help of software agents	65
6.4	Platform Capabilities and Constraints	66
6.5	Usability Goals	66
6.6	Work Model	66
6.7	Conclusions	67
6.7.1	Evaluation	68
6.7.2	Open issues	69
6.7.3	Implementation	69
7	Wiki-based ontology population	71
7.1	MediaWiki and Wikipedia	71
7.2	Semantic extensions to MediaWiki	72

7.2.1	The Big Picture	73
7.2.2	Usage of Typed Links	74
7.2.3	Attributes and Types	75
7.2.4	Semantic Templates	77
7.3	Design	78
7.3.1	Typed Links	79
7.3.2	Attributes	80
7.4	Implementation	83
7.4.1	User Experience	83
7.5	Usage scenarios	84
7.6	Conclusions	86
8	Outlook	89
8.1	Siemens Case Study	89
8.2	Legal case study	90
8.3	Digital Library Case Study	90
9	Conclusions	93
A	Description of EvA screens	99
A.1	Screen layout conventions of the eclipse platform	99
A.2	Screen design	100
A.3	General Design Principals	100
A.4	Requirements not regarded	102
A.5	Description of screens	102
A.5.1	Main argumentation window	102
A.5.2	Dialog “Create new concept”	112
A.5.3	Dialog “Select sources”	113
A.5.4	Dialog “Select issue”	114
A.5.5	Dialog “Select person”	114
A.5.6	Dialog “Select concept”	114
A.5.7	Dialog “Select instance”	115
A.5.8	Dialog “Task X”	116
A.5.9	Dialog “Filter tasks”	117
A.5.10	Dialog “Search”	118
A.5.11	Dialog “New issue”	119
A.5.12	Dialog “New source”	120
A.5.13	Dialog “Watchlist for issues”	121
A.5.14	Dialog “New argument”	122
A.5.15	Dialog “New elaboration”	122
A.5.16	Dialog “New relation”	123
A.5.17	Dialog “New attribute”	124
A.5.18	Dialog “New instance on concept X”	126

A.5.19 Dialog “Welcome”	127
A.5.20 Dialog “Import text sources”	128
A.5.21 Dialog “Welcome to project X”	129
A.5.22 Dialog “New idea set”	130
B EvA glossary	131

Chapter 1

Introduction

1.1 Motivation

Progress in the development of the Information Society has seen a truly revolutionary decade, with the advent of widespread networked information services. A new social revolution is imminent, involving the transition from Information Society to Knowledge Society. SEKT aims to position the EU IT industry at the core of this, by developing the essential semantic knowledge technologies for realizing the European Knowledge Society.

The vision of SEKT is to develop and exploit the knowledge technologies which underlie Next Generation Knowledge Management. We envision knowledge workplaces where the boundaries between document management, content management, and knowledge management are broken down, and where knowledge management is an effortless part of day to day activities.

SEKT is built around the synergy of the complementary know-how of the key European centres of excellence in Ontology and Metadata Technology, Knowledge Discovery, and Human Language Technology. The execution of SEKT is based on the integration of fundamental research, component development and integration driven by real world case studies in the public and private sectors. SEKT will provide new insights on knowledge technologies.

The main objective of work package 7 is to provide a methodology for the application of Semantic Knowledge Technologies into enterprises in a goal-driven and process-oriented way. Hence we worked on an integrated approach balancing the organisation and management aspects on the one hand, and the information technology and systems aspects on the other hand. We developed a methodology which integrates specific issues of the three core technologies that are explored within SEKT. Particularly, we address the issue of combining manual and semi-automatic approaches and when to use which combination of the SEKT technologies [TSVP04]. We call the developed methodology

DILIGENT [STV⁺05].

Central to Semantic Knowledge Technologies is the application of ontologies to knowledge management problems. Core aspects for the methodology therefore include the efficient and effective creation and maintenance of ontologies in settings such as provided by the case studies. Instead of describing the technologies and the progress in the case studies in detail we focus on gathering the lessons learned from having applied the technologies in the case studies, as a first step towards incorporating recommendations and best practises into the methodology, which will be part of next year's final deliverable of this work package.

1.2 The SEKT Big Picture

This report is part of the work performed in work package (WP) 7 on “Methodology”. As shown in Figure 1.1 this work is closely related with “Usability and Benchmarking”; they both are intermediating between the research and development WPs and the case study WPs in SEKT.

The case studies present their work in their own deliverables, namely [SEK05f, PBC⁺05, SEK05a]. Here we focus on the lessons learned and of interest to a general audience, providing a description of the case study only to give context. Also, the technical work packages have deliverables of their own, describing in far more detail the new results of their technology (for example, [SEK04, SEK05c, SEK05e], and many others). Here we do not describe the technologies, but we focus on gathering the lessons learned from having applied the technologies in the case studies.

This report describes the initial application of the SEKT methodology and technologies in the case studies. As the case studies are scheduled to start in the third year, this remains a preliminary report. It focusses on the case studies, where we could already gain insight in the working of the methodology. Second, it describes how to capture the methodology with the existing or newly developed SEKT tools. Issues not captured by the other SEKT work packages are handled in detail and specific tools are described.

1.3 Organization of this deliverable

This deliverable consists of two major parts: in part I we describe the experiences and lessons learned from the different technology sectors SEKT is combining. Although some of this has been published in other places, here we summarize the lessons learned from the SEKT work packages in one place. As this is a best practises and lessons learned deliverable, we capture and distil the experience of the whole SEKT project here. In chapter 2 we will take a look at how the Knowledge Discovery and Machine Learning expertise was used inside the SEKT project. In chapter 3 we write about the Human

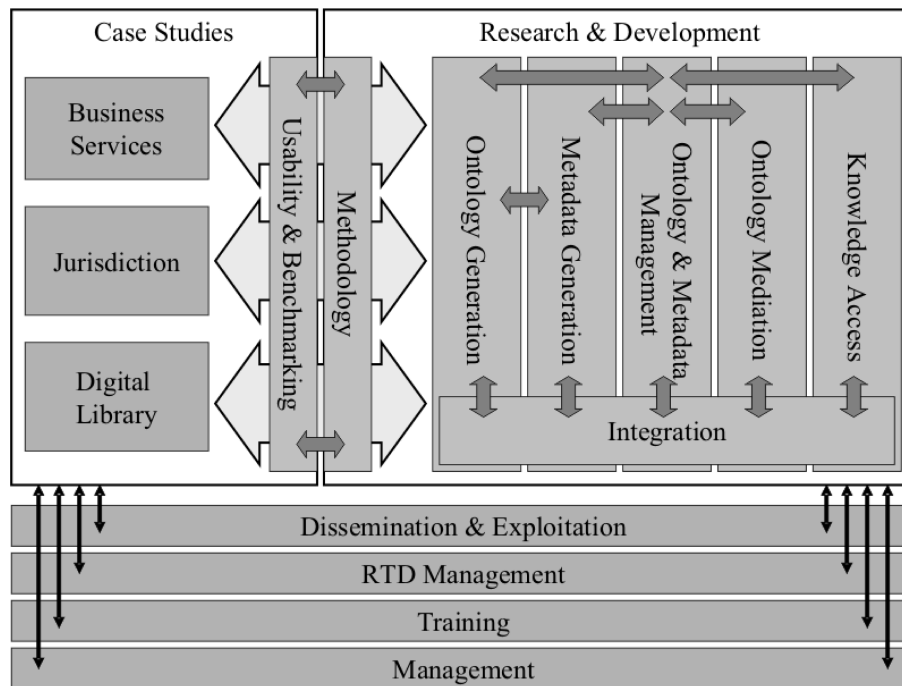


Figure 1.1: The SEKT Big Picture

Language Technologies applied to the case studies. Then, chapter 4 describes the experiences and problems we encountered when using Ontologies and Ontology Management technologies. Chapter 5 will present the extensive experience we gained from applying DILIGENT to the case studies. These four chapters provide a short description of the applied tools and methodologies, and of the lessons learned within the case studies.

The second part of this report delivers the description of the design of two tools that follow the DILIGENT methodology. Inside the case studies we have used several tools and tool combinations to support the users. This highlighted numerous problems with the existing tool infrastructure. Based on our methodology and the experiences described in the first part, we have designed two tools to support knowledge workers dealing with ontologies. In chapter 6 we present the tool design for EvA, which helps with the creation, maintenance and evolution of an ontology. Chapter 7 talks about a wiki-based infrastructure used for the distributed population of an ontology.

In chapter 8 – Outlook we present our plans for the third year of the SEKT project, where the use cases will be up and running and – hopefully – the tools described in the second part of this report will have found a wide audience. Finally, the conclusions in chapter 9 will summarize the most important lessons learned.

Part I

Lessons learned

Chapter 2

Knowledge Discovery

This chapter will examine the process and practice of knowledge discovery as a part of the SEKT project. The primary realization is how knowledge discovery technologies are to be linked with knowledge engineering and other techniques into a coherent process. Ontology engineering is very similar in nature to knowledge discovery: both tasks result in codified and formalized knowledge (e.g., in form of ontologies). But there are important differences: knowledge engineering relies on the experience of knowledge engineers and domain experts. Knowledge discovery, on the other hand, requires a large-enough corpus of data, which must be cleaned and prepared. Subsequent tasks involve providing the right view of the data, where irrelevant information is filtered out, and important information is accentuated.

Knowledge discovery has contributed to two crucial aspects of the case studies:

- Semi-automatic ontology learning summarized the information in a large corpus of text and data. The summaries are not yet ready for consumption, as domain experts may disagree with the proposed concepts. Still, it is impossible to manually analyze gigabytes of raw text or web logs.
- Given an annotated corpus of data, knowledge discovery is capable of learning how to annotate automatically. This is of considerable utility, as the computer is now capable of the automatic detection of concepts based on the human point of view. Of course, the ontology must be designed in a way that allows this automatic annotation.

We will first examine the knowledge discovery process. We will then focus on the problem of preparing, filtering and pre-processing the data. We will continue by showing how the the problem of ontology evaluation can be addressed by the technique of classification.

2.1 The Knowledge Discovery Process: Best Practices

Knowledge discovery is a key technology in its own right for knowledge management, providing knowledge workers with summarized information, extracted from large volumes of data. At the same time, like natural language processing, knowledge discovery (KD) has a role in metadata extraction to enable the automatic or semi-automatic mark-up of data, ontology learning and evolution. Obviously, KD interacts closely with human language technologies for e.g. metadata generation, and ontologies provide the background knowledge for improving KD results.

In order to fully benefit from these technologies, they must be used together. This convergence is now timely because of the maturing of the 3 separate disciplines, particularly ontology technology, which has received much attention over the last 2-3 years. Thus, in SEKT, knowledge discovery techniques will provide ontologies based on the information obtained from text with natural language technologies. Furthermore, metadata will be extracted automatically from the text, again with the help of natural language technologies. These ontologies and their corresponding metadata will be managed and evolved using enhanced ontology and metadata technology developed in SEKT.

Ontology generation, addressed in SEKT WP1, provides different knowledge discovery methods contributing to semi-automatic ontology construction. There are several definitions of the ontology engineering and construction methodology, mainly putting emphases on knowledge management perspective. For instance, the DILIGENT ontology engineering methodology [VPST05] that defines five main steps of ontology engineering: building, local adaptation, analysis, revision, and local update. Phases of semi-automatic ontology construction are defined in [GM06] analogous to the CRISP-DM methodology defined for the knowledge discovery in general [She00]. ¿From the perspective of knowledge discovery, semi-automatic ontology construction can be defined as consisting of the following interrelated phases:

1. domain understanding (what is the area we are dealing with?),
2. data understanding (what is the available data and its relation to semi-automatic ontology construction?),
3. task definition (based on the available data and its properties, define tasks to be addressed),
4. ontology learning (semi-automated process addressing the defined tasks),
5. ontology evaluation (estimate quality of the solutions to the addressed tasks), and
6. refinement with human in the loop (perform any transformation needed to improve the ontology and return to any of the previous steps, as desired).

If we related this phases to the stages in the methodology defined for manual construction of ontologies [UK95], we can see that the first two phases referring to domain and data understanding roughly correspond to the first manual stage of identifying the purpose of the ontology. The next two phases (tasks definition and ontology learning) correspond to the second manual stage of building the ontology. The last two phases on ontology evaluation and refinement correspond to the third (final) manual stage of evaluation and documentation. More on using different knowledge discovery methods through theses phases can be found in [GM06]. In the rest of this Chapter we focus on the ontology evaluation phase in the context of SEKT case studies.

2.2 Getting and Preparing the Data

2.2.1 Availability of Data

Probably the single most important issue was the availability of data. This was problematic in all the case studies. In the legal case study alone (see section 5.2, or, for a more detailed description, [CPC⁺04]), gathering the questions for the intelligent FAQ took an ethnographic campaign that went for months and consisted of lots of interview throughout Spain (see also section 5.2). Crawling the text corpus of judgements for the second part of the prototype, to gather the data took approximately another three to four months, checking the process a couple of times every day, and working during the holidays and weekends when the server was under less load. And once the data is available, it still needs to be preprocessed.

Also the data is usually not available in a specific format, one is happy to get the data in any way. This was particularly evident in the user profiling part (D5.2.1) of the SEKT BT case study [GMG05b], where Web logs of accesses to a digital library first needed to be extensively preprocessed (details in [GMG05a]) and then aligned with the ontology of the digital library documents. Filtering the data set according to the criteria required for the tools (i.e. identifiable user accounts with enough history) resulted in having only 0.22% of the initial database size left, leading to an extremely sparse dataset with information collected implicitly on the server side. This significantly hurt the performance of the collaborative filtering methods applied on the data for user profiling.

Not all data sources were controlled by project partners, so we had to invest time in the required data processing. Even if they are controlled by the project partners, providing the data can be a very complicated and formal procedure that may need to be repeated based on findings during data pre-processing and alignment.

Sometimes, even handing over the data can be surprisingly complicated. Although the problems listed here are not part of any technical constraints, we deemed them necessary to mention, as the context of the SEKT methodology covers the whole range of introducing SEKT technology within an enterprise setting. Underestimating such problems will

easily lead to missed deadlines and late deliveries.

2.2.2 Preprocessing the Data in the Legal Case Study

The data, i.e. the judgement corpus for the second part of the prototype, had to be prepared in order to enhance precision and recall. The knowledge discovery process treats each term as an appearance of a primitive concept. In subsequent operations we infer similarity between two documents from a similar number of appearances of the primitive concepts. If the primitive concepts are irrelevant, the similarity will be arbitrary.

Primitive concepts are usually simply words. But words do not correspond directly to concepts: some words carry no relevant meaning, while others may be important (or key). Some words have multiple meanings, but multiple words may also have the same meaning. For that reason, two main operations in data preparation are: to unify all the words with a very similar meaning, and to remove the words that do not carry relevant meaning. The first operation is achieved using stemming, lemmatisation and with thesauri (including WordNet). The second operation is performed by the domain experts, who remove the words that are deemed not to carry any relevant meaning for the task of determining sub-domains and topics. Such words (or lemmas) are referred to as *stop words*.

In stemming, we remove the final few letters of the word that carry grammatical nuances but do not distinguish the meaning. For example, we represent words *dog*, *dogs*, *doggy* with *dog+*. A verb such as *be* may take several forms: *is*, *are*, *were*, *being*, *be*. During lemmatisation, we replace all these forms that may reflect tense, gender, inflection, irregularity, etc., with the simple canonical one: *be*. While this particular example may seem contrived, other languages, such as Spanish, have many more such examples. Such languages benefit much more from lemmatisation, while English does well already just with stemming [VMF⁺05].

Words to be Preserved in the Corpus

In general terms, the words that contribute most to the meaning of a text are nouns and verbs. Besides the stop words, these kind of words have to be preserved in the corpora in order to perform a good search. The codes are:

1. For verbs (main verbs): VM—
2. For nouns:
 - (a) Common nouns: NC—
 - (b) Proper nouns: NP—

In reference to the other categories not mentioned so far (adjectives, adverbs and abbreviations), we think they should be preserved for some reasons. First, adjectives (AQ—)

and substantives (NC— and NP—) —in a broad semantic sense— are considered to form a joint category that some authors call ‘noun’, which means that the adjective has not always been treated as a completely independent category, for it works as a complement for the sense of the substantive. To delete it would imply a great loss of information.

A typical example of this would be *malos tratos* [ill-treatment], which is formed by an adjective [*malos*] plus a substantive [*tratos*]. Secondly, however its strength and importance in sense-adding value is not the adjective’s (at least in terms of occurrence), adverbs (RG—) are still important for our purposes, because they may add important information about the use (thus the sense) of a verb in many cases. An example of this may be *tramitar urgentemente* [to proceed urgently]. Thirdly, abbreviations (Y) (like *etc.*, but also like *LeCrim* [abbreviated form for referring to the Criminal Code]) are very important for our purposes, since the legal discourse makes a great use of them and can be of great value for classification.

Domain Dependent Words

The Ontology of Professional Judicial Knowledge (OPJK), developed for the iFAQ, currently covers the sub-domains of *guardia* [on-duty period], gender violence and procedural law. Currently, it has to be completed with issues from criminal and commercial law. Obviously the words —whether they are concepts or instances— which appear in this ontology ought to be especially taken into account when considering terms in the corpus of judgements. In other words, OPJK ought to be seen (for this purpose) as a statement of relevant linguistic terms that are important for capturing the semantics we care about. This ontology is being built from textual protocols extracted from an extended field work, which *per force* implies a strong relation between words and real data. Moreover, the extraction of the words (concepts and instances) has been made manually, mostly extracted by hand by the legal expert team. Thus the words which appear in OPJK ought to be given some special consideration.

Moreover, as a good number of the terms in OPJK do not correspond to a single word but to a compound form, these compound forms ought to be taken into account as single occurrences and not as two. Typical forms extracted from OPJK are:

- Noun + Preposition + Noun (*juzgado de guardia*, *diligencia de embargo*, etc.)
- Adjective + Noun (*malos tratos*, *lesa humanidad*, etc.)
- Noun + Adjective (*documento procesal*, *enjuiciamiento civil*, etc.)

2.2.3 Lessons Learned

Probably the most important factor for the success of deploying SEKT knowledge discovery technology is the availability of data, the quality of the data, and the sufficient quantity

of the data. It is also important to understand the requirements of the technology. For example, to detect a concept reliably from text using statistical methods, there must be at least about 50 examples (but ideally more than 100), and 100 or more counter-examples.

The analysis has demonstrated the need to treat the corpus prior to semi-automatic ontology construction to assure better results. This was taken into account in the next phase where OntoGen [SEK05b] was used for semi-automatically construction of a topic ontology of questions. Actually, in legal cases study two topic ontologies were created using OntoGen, one for the questions corpus and one for the judgements corpus, the latter still in development. For both corpora, preprocessing involved lemmatization and removing of stop words. For the judgements corpus, a document was actually a concatenation of all the summaries that correspond to a particular primary topic. This contributed to efficiency of semi-automated ontology construction.

In summary, the following lessons are to be learned:

- the data must be available in sufficiently large quantities before the knowledge discovery technologies can take place.
- collecting the data can be expensive and legally problematic, and it is not guaranteed that the quality will be sufficient.
- preparing the data involves both knowledge discovery experts, and domain experts. In the case of knowledge discovery from text, some experience with natural language processing is required.

2.3 Using Knowledge Discovery for Ontology Evaluation

Knowledge discovery methods can be used for evaluation of ontologies needed for different reasons including comparison of several ontologies, guiding the ontology construction process, evaluating semi-automatic approaches to ontology construction. Approaches to ontology evaluation are applied on different kind of ontologies and for different purpose. Most approaches fall into one of the following categories [BGM05b]: approaches based on comparing the ontology to a “golden standard” (usually created manually), approaches based on using the ontology in an application and evaluating the results, approaches involving comparisons with a source of data (e.g. a collection of documents) about the domain that is to be covered by the ontology, approaches where evaluation is done by humans who try to assess how well the ontology meets a set of predefined criteria, standards, requirements, etc.

Evaluation of ontologies is an important aspect of construction and usage of ontologies and knowledge management in general, thus it is addressed in SEKT through several perspectives in the corresponding deliverables as follows.

[BGM05a] proposes an approach to ontology evaluation primarily geared to enable automatic evaluation of an ontology that includes instances of the ontology concepts. The approach is based on the golden standard paradigm and its main focus is to compare how well the given ontology resembles the golden standard in the arrangement of instances into concepts and the hierarchical arrangement of the concepts themselves. Its main difference from the other existing approaches is basing the evaluation on instances assigned to the ontology concepts without relying on natural-language descriptions of the concepts and instances, or using relaying on the representation of instances. What is needed is only that we can distinguish one instance from another and that the ontology is based on the same set of instances as the golden standard.

In WP3, D3.2.1 [VVS05], there is a cost associated with an ontology. By decreasing the cost, we can improve the ontology. The cost depends on the patterns of usage, represented by a set of queries. A good ontology should facilitate frequent queries better than rare queries. Furthermore, the cost increases with the length of the path from the root to the correct concept. Concepts with a very large number of subconcepts also increase the cost, as it is harder to choose the right path when there are many choices available.

In WP10, D10.3.1 [PBC⁺05] knowledge discovery techniques are used to annotate previously unseen documents in order to evaluate the ontology against the documents in the process called *ontology grounding* [JM05]. The documents are annotated by topic categories that are part of a topic ontology. In this way, each topic can be analysed based on the documents that were automatically annotated by that topic and if a topic turns out not to be grounded, there are several ways of remedying the problem:

- An increase in the amount of training data.
- Redesign or simplification of the topic ontology.
- An increase in the amount of background knowledge:
 - Addition of features that recognize phrases.
 - Specification of feature importance.
 - Removal of features that are known to be irrelevant or that cannot be used by the classifier.

It is usually insufficient to merely analyze the classification performance of a model. To diagnose the reasons and courses of remedy, we need to examine the problems in more detail. Specifically, we should understand the causes of the errors. Such causes will provide insight about the missing background knowledge which can be introduced manually. To facilitate the analysis, in WP10 two techniques were employed: identifying the decisive features for each concept and identifying the instances where the concept is not identified correctly.

2.3.1 Evaluation of Ontologies in the Legal Case Study

In SEKT legal case study, the competency questions were used to evaluate the manually constructed Ontology of Professional Judicial Knowledge (OPJK). This ontology was used as the underlying semantic model to compute the input question with natural language processing software (tokenizer, morphological analyser, basic shallow parser and semantic analysis) for retrieving the question-answer pair of FAQ stored within the system [CPC⁺05]. The process of extraction of concepts from the text in order to both model OPJK and show the importance of the source of the data (text) when constructing ontologies of legal professional knowledge is largely explained in chapter 5.

Here we focus on the use of this same corpus of questions to automatically model a second ontology, this time a topic ontology, to allow legal sub-domain detection. The sub-domain detection helps with search: if we can detect the topic of a query (such as *gender violence*), we can offer answers that cover that particular topic. Topic detection does not require the user to be familiar with a specific nomenclature of topics.

First, we will introduce topic ontologies and the manually developed classification into topics (or legal sub-domains) by the legal expert's team. Then, we will describe the analysis of the results provided by a classifier, which results in recommendations on how to prepare the corpus prior to the statistical analysis of the text. A more thorough description of this procedure, focusing more on the tool used and the technical details, can be found in [PBC⁺05].

Legal question topic ontology

A topic is a semantic annotation of a document that people instinctively use to search and organize information. While we can semantically annotate many facets of a document, the notion of topic covers the document as a whole: it is the conceptualization of the document with the greatest breadth. When we seek information, our primary intention is usually to discover information on the same topic as is the topic of our problem. Topic can be distinguished from other descriptors of a document, such as language, style, punctuation, grammar, reputation, graphical design, author, date, and so on.

Initially, the team of domain experts identified several sub-domains (topics) intuitively gathered from the questions. The detection of sub-domains drove, at the beginning, to a classification of questions into these different sub-domains to ease the construction of the ontology, which focused, at first, on two specific sub-domains: *on-duty* and *family issues* (containing *gender violence* and *protection* or *restraining orders*). The manual classification of questions into sub-domains resulted in the list shown below:

- | | |
|---------------------------------|---------------------|
| 1. On-duty | 7. Court office |
| 2. Family issues | 7.1. Organization |
| 2.1. Gender violence | 7.2. Civil servants |
| 2.2. Minors | 7.3. Prosecution |
| 2.3. Divorce/separation/custody | 7.4. Infrastructure |

- | | |
|-------------------------------|-------------------------------|
| 2.4. Other | 7.5. Security |
| 3. Immigration | 7.6. Informatics |
| 4. Property | 7.7. Experts |
| 5. Sentencing | 7.8. Police and security |
| 5.1. Execution | 7.9. Incompatibilities |
| 5.2. Breach of order/judgment | 7.10. Relations with the GCJP |
| 5.3. Form | 7.11. Other |
| 5.4. Notification | 7.12. Lawyers |
| 6. Proceedings | 8. Business Law |
| 6.1. Evidences | 9. Traffic accidents |
| 6.2. Competence | 10. Criminal Law |
| 6.3. Criteria | 10.1. Drugs |
| | 10.2. Organized crime |
| | 10.3. Theft/Robbery/Burglary |

Later on, during the manual conceptualization of OPJK, the domain experts and ontology engineers in the team realised that most of the questions regarding *on-duty* and *family issues* were necessarily related to procedural moments (phases or acts, *proceedings*). Moreover, *decision-making and judgements* were also necessarily related to *proceedings* and *gender violence* (included within *family issues*) was necessarily related to *Criminal Law*, thus related to the other sub-domains being modelled. This fact showed, again intuitively, that the questions could have been misclassified and also that the questions could refer to more than just one intuitive sub-domain at the same time.

This intuitive conclusion was confirmed by the results provided by a support vector machine classifier. In simple terms, the classifier attempted to learn by which keywords to distinguish the topics or sub-domains of questions. Ideally, the classifier would automatically, correctly and unambiguously decide what sub-domain a particular question belongs to. In practice, however, the classifier will make mistakes. Some of the mistakes might indicate that the categories are not specified with a sufficient amount of detail. Other mistakes may indicate incorrectly labelled questions, but there might also be questions that can be correctly classified in several ways.

The classifier first analysed the intuitive classification in sub-domains and provided us with two different statistical reports. The first report was a list of keywords for each sub-domain. The list of keywords showed which terms the classifier thought to be in favour or against of that sub-domain. The second report was a diagnosis for each of the intuitively and manually classified questions, listing arguments for or against that classification.

Analysis of grounding of a topic

It is not possible to discover sufficiently good rules from a limited set of data. We have employed the state-of-the-art SVM classifier [Joa98] and checked if a particular topic can be detected. To properly evaluate a topic detector, however, we need to apply it on the data that was not used for training the classifier: like a student it could simply memorize the right answers instead of comprehending the meaning of the category. The technique we employed is 10-fold cross-validation. We split the questions into 10 groups of approximately equal size. Now we perform 10 experiments, where one group is hidden

away, and the other 9 are used to train the classifier. The resulting classifier is then checked on the hidden test data. Because the results may depend on the allocation of questions into groups, we perform cross-validation twice, with two different ways of partitioning into groups.

There are numerous ways of evaluating the classification performance. The most popular method is classification accuracy: in what proportion of the test data was labelled correctly. This, however, is not optimal when a topic is somewhat rare: the high accuracies may be misleading - if a topic appears in 1% of all documents, we get 99% classification accuracy by simply denying the existence of the topic. Because of that, the information retrieval community uses the measures of precision (out of the questions classified as having the topic, how many really had it?), and recall (out of the questions having the topic, how many were tagged by the classifier?). F-score combines both precision and recall in a single number. The results obtained on the question corpus are as follows:

name	#	CA	PRE	REC	F
GUARDIA	79	94.4	0.78	0.63	0.70
FAMILIA	103	92.9	0.82	0.61	0.70
PROCESO	307	75.1	0.69	0.68	0.68
MERCANTIL_NEGOCIOS	28	97.9	0.79	0.59	0.67
SENTENCIA	48	96.2	0.79	0.52	0.63
EXTRANJERIA	26	97.5	0.67	0.50	0.57
OFICINA_JUDICIAL	151	85.1	0.66	0.48	0.56
ACCIDENTES_TRAFICO	11	98.9	0.67	0.45	0.54
PENAL	10	99.0	1.00	0.20	0.33
PROPIEDAD	10	96.4	0.16	0.40	0.23

Here, # denotes the number of examples of that topic in the corpus; altogether there were 773 questions in the corpus. It is also important to note that a question can belong to multiple topics. An example of this are topics and subtopics: an example belongs to all the supertopics of a topic. But it is also technically possible to annotate a question with several topics that are not nested: a question that combines criminal and family law would belong to both topics.

We can see that rare topics (*penal* [criminal law], *propiedad* [real estate]) were very difficult to detect, in part because there were only 10 examples: such topics are not grounded well. On the other hand, *familia* [family] and *mercantil-negocios* [business] were detected reasonably well even if the number of instances was not that large: they are grounded well. As a side note, through detailed analysis we also found that *guardia* [on-duty period] and *proceso* [proceedings] were often confused.

Of course, the results are not very good by the standards of text classification. To improve them, we would need a larger corpus of questions (additional examples) and additional background knowledge (which words to include, which words are important).

To understand the reasons for mistakes, we have performed detailed analysis of the errors, which will be now described.

Detailed analysis of misclassifications

Because all the questions have been manually classified by the experts, we know what is the true classification of a question. If the question is then shown to the classifier, we can check if the automatic classification is consistent with the manual one. If there has been a mistake, we can examine what features were decisive in the classification. We show five features with the highest and five features with the lowest leverage for the wrong classification. Furthermore, we do the same for the correct classification.

The errors may arise for four reasons: overvaluing of arguments in favour of wrong decision and against the correct decision, and undervaluing of arguments in favour of the right decision and arguments against the wrong decision. In practice, the classifier might not detect an important feature, or might detect an irrelevant feature.

When this analysis was related to the analysis made by the same domain experts of the results of the diagnosis for each of the questions originally classified in the on-duty sub-domain, they could detect that:

1. Some on-duty questions had been misclassified in that sub-domain and they clearly belonged to another.
2. Some on-duty questions were correctly classified into the on-duty period but were also relevant to other sub-domains.
3. Some true on-duty questions were shown wrongly to belong to another sub-domain.

An example of such analysis is as follows:

Qué hago con el procedimiento cuando mando análisis de ADN y tardan casi un año en llegar los resultados?

True=guardia Predicted=proceso/pruebas

* Keywords in favor of guardia		* Keywords against guardia	
0.00 (0)	estar	1.35 (4)	mandar
0.00 (0)	juzgar	0.36 (6)	llegar
0.00 (0)	unico	0.06 (6)	ano
0.00 (0)	guardia	0.04 (1)	?
0.00 (0)	permanente	0.00 (4)	procedimiento
* Keywords in favor of proceso/pruebas		* Keywords against proceso/pruebas	
2.99 (6)	analisis	0.32 (4)	procedimiento
1.85 (7)	casi	0.04 (1)	?
1.85 (7)	adn	0.00 (0)	estar
1.29 (6)	ano	0.00 (0)	juzgar
1.11 (6)	resultado	0.00 (0)	unico

We can see that *mandar* is an argument against the *guardia* topic, while *analisis* is an argument for *proceso/pruebas* topic.

In conclusion, the data showed that the classifier encountered several problems in classifying the questions, mostly due to the inefficient original classification. One of the reasons was automatic misclassification. However, there was another reason: there was the need to consider the on-duty sub-domain differently. The on-duty sub-domain does not represent a conceptual domain of specific knowledge, it should represent a time domain. The on-duty period involves all those questions that happen on a certain time. Therefore, the questions belonging to the on-duty period could and should belong also to the other sub-domains that represent conceptual domains and contain the questions in a certain knowledge area.

Secondly, the classifier misunderstood the relevance of some of the keywords used for argumentation. As stated above, some terms were not relevant for a particular sub-domain. However, the main difficulties were:

1. The usage of complete irrelevant terms for the legal domain (at least on their own) like but [*pero*], to order [*mandar*], to follow [*seguir*], to put [*poner*], that [*ese*], work [*trabajo*]);
2. The inability to capture relevant compound forms. For example: bad/ill [*malo*] and treatment [*trato*] are regarded separately, while they are actually a compound form: bad/ill treatment [*malos tratos*] is a relevant key-phrase, used to identify the questions related to gender violence.

Therefore, a conceptual change was adopted and on-duty was regarded, from that point, as a time domain, rather than a sub-domain in itself.

2.3.2 Lessons Learned

The development of the automatically learned topic ontology started with a manually modelled version of the topic ontology, that was soon disregarded by the evidence given by the classifier. The automatic classification of a set of questions taking into account the previous manual classification resulted in misclassification of questions, and low relevancy of sub-domain keywords. We have shown usual classes of misclassification, and we have provided an automatic evaluation method for the original topic ontology. This revealed errors in the manually created topic ontology, which could be resolved by analysing the computer generated arguments.

In summary, we have learned several lessons:

- when working with text, one has to mind the semantics of individual words: some words are relevant, others irrelevant; some words have multiple meanings, some

meanings have multiple words. The words have to be mapped into meaningful primitive concepts.

- when working with concepts, one has to mind the association between concepts and words. This is achieved by examining the overall classification performance, and by investigating the causes of classification mistakes.
- if the goal is effective automatic classification, the ontology must be designed in a way that allows this. The classifier can provide feedback to the domain experts and knowledge engineers. When a classifier is having problems with an issue, a human being might have problems too.

Chapter 3

Human Language Technologies

In the SEKT project Human Language Technologies (HLT) have two key contributions that will both include the handling of multilinguality (see Fig. 3.1). In the first place it will be used to semantically annotate informal and unstructured knowledge. Thus, the automatic or semi-automatic extraction of metadata from legacy data will be achieved. Secondly natural language processing will be used to generate natural language based on formal knowledge (ontologies and metadata). Here, HLT will be strongly integrated with methods from KD and OMT. The ontologies that structure metadata are in many cases language-independent to a significant degree. SEKT will trial metadata generation methods based on Information Extraction, Content Extraction and other language analysis technology that is used in HLT for various languages and, similarly, prove the documentation of ontologies and metadata in practice (using Natural Language Generation).

All the HLT technology used and further developed in the project will be based on systems that have been proven in a large range of languages. For example, the extraction components were recently entered in the “TIDES Surprise Language Competition” (that measured the ability to port HLT systems to Hindi) with favourable results. All the main European languages, and others from Chinese to Bulgarian, have also been covered in various ways, and SEKT will include at least four languages directly in the case studies (English, German, French and Spanish).

3.1 Using HLT in End-User Applications

Full-depth automatic understanding of free text is beyond the reach of current natural language processing (NLP) applications and is most likely not achievable in the foreseeable future. However, experiments has shown that in many cases certain characteristics of the documents (such as genre, style, subject, layout, etc.) are known to be fixed, which allows some automatic analysis to be successfully performed.

The task of harvesting structured, quantifiable and unambiguous data from natural

language text is known as Information Extraction (IE) [CL96, GW98, App99, Cun99]. Various types of data may be extracted, starting from simple entity identification, through entity reference resolution (i.e., determining which entity a pronoun such as ‘he’ refers to), to the identification of complex events and relations. The outputs of IE are records similar to database entries which can be used for a wide range of operations such as data mining, automatic summarisation or report generation.

In the last few years IE has become a highly productive research area in the NLP field, a large number of R&D sites – both academic and commercial – being involved in the technology.

So far, IE has proved valuable in a number of application contexts, including:

competitor intelligence: e.g. tracking prices offered by competing on-line services [FSW01];

summarisation: e.g. extracting the most significant sections from long company reports referring to health and safety issues [MBS⁺02, LR02];

word processing: e.g. the ‘smart tags’ facility in recent versions of MS Word;

e-science: e.g. extracting plant taxonomy data from biological texts in order to compare and synthesise varying approaches [WLT⁺03];

multimedia indexing: e.g. analysing football commentaries, news articles and web pages relating to football matches, in order to index and semantically annotate videos of the matches [SCM⁺02, SCB⁺02];

digital libraries: e.g. marking-up 18th century criminal trial reports for humanities researchers [BMCS02, BCT⁺02].

on-line collaboration e.g. performing textual analysis by users at several geographically disjoint locations on a shared collection of texts [TBMC03].

IE systems are labour intensive to develop, requiring the expertise of highly skilled personnel. Currently, they are only cost effective when used for large volumes of textual data and for static information needs.

The changing nature of the information need and the diversity of application areas creates a demand for IE that can be quickly and easily ported to new tasks. Similarly, the emergence of the semantic web, where much of the textual information currently present on the Internet will need to be enhanced by the addition of semantic annotation, creates a need for portable IE: “due to the vast amount of data that will need processing, domain experts cannot be expected to create the annotations manually” [CT02]. Portable IE systems can be adapted for creating semantic annotation for sets similar web pages using far less resources than would be required to create the annotation manually.

To make IE useful where there are dynamic information needs or smaller text volumes requires the *portability* of IE from one application to another to be maximised. Below we summarise the multiple dimensions of IE portability:

task: different tasks involve different combinations of IE data types of varying complexity;

genre and style: the language involved varies from formal and precise (e.g. news articles) through informal and badly spelled (e.g. emails);

formats: from MS Word to XML, HTML, etc.;

encodings: many different character sets are in active use around the world;

domain: the subject matter of the texts to be analysed;

scale: IE applications vary from the small scale to the size of the web itself (i.e. huge scale), and processing requirements vary from overnight to near real time;

robustness: a general requirement is that the process should be robust in all cases;

end-user type: some users may have a sophisticated knowledge of computing and/or linguistics, whereas others may have almost none.

The GATE HLT¹ infrastructure which is at the core of SEKT already provides support for some of these issues, so they do not need to be addressed separately in each application. For instance, support for different document formats and encodings is covered extensively [TUB⁺02].

The remaining portability issues need to be addressed individually during the design and development phase of each application, however, GATE provides tools to support this process. Generally speaking, application development goes through several phases, where the second and third one can be iterated over several times:

1. data collection, annotation, and analysis
2. modification of general purpose HLT components to domain specific data
3. performance evaluation
4. deployment within end-user application

The data collection, annotation and analysis phase involves the creation of a corpus of documents and its manual annotation by end-users. This process is supported by GATE via two main components: the multilingual document editor (Fig. 3.1, see [TUB⁺02]) and the ANNIC corpus analysis tool (Fig. 3.2, see [ATBC05]).

¹<http://gate.ac.uk>

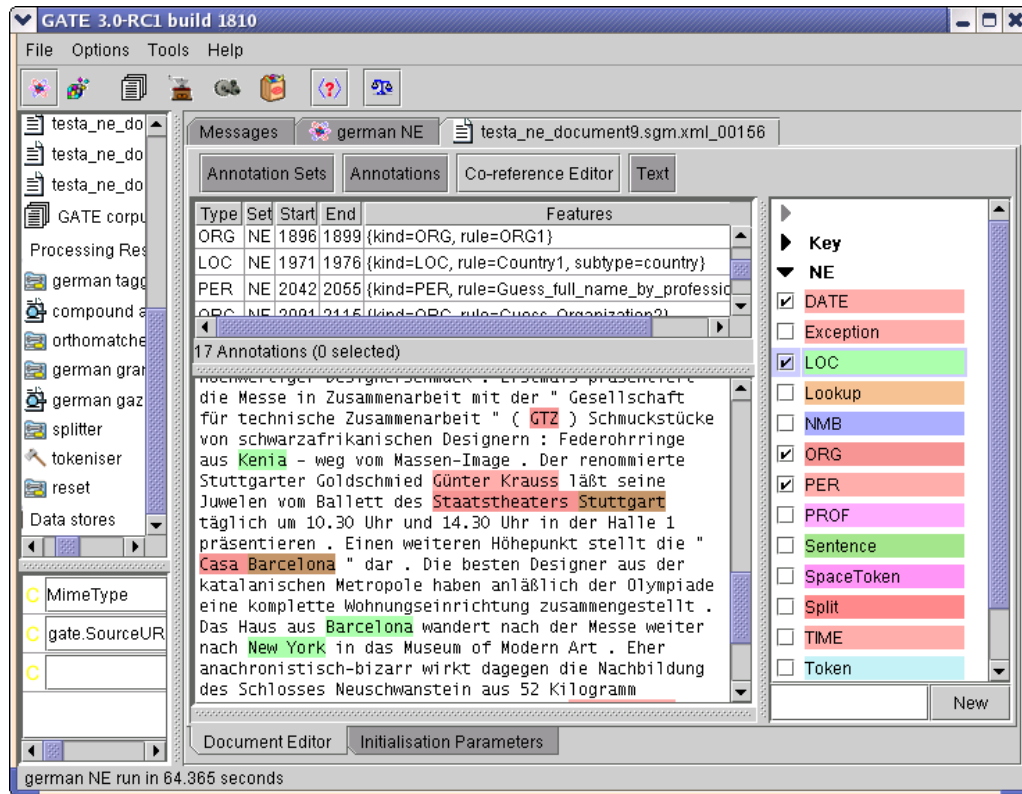


Figure 3.1: GATE's multilingual document editor

The GATE development environment supports the process of modification of the HLT components to domain specific data in a number of ways:

- the gazetteer list editor enables users to modify and extend the lists that provide domain knowledge to the system. These lists can in fact be derived from the ontology [BTMC04] or be directly connected to an ontology repository (see deliverable [SEK05c] on Massive Semantic Annotation). In addition, the annotated data can be used to gather automatically new entries via the gazetteer list collector [MBC04].
- the lexicon of the part-of-speech tagger and the grammars of the semantic tagger can be modified easily with a text editor and re-loaded and tested within the system. Guidelines for easy adaptation to new languages have been provided [MTBC03] and SEKT deliverable [SEK04] on Language Issues.
- the tokeniser, sentence splitter, and orthographic co-reference module are largely reusable without modifications across many European languages. Minor modifications are easily carried out, if necessary (see SEKT deliverable [SEK04] on language issues).

An essential part of the application development process is performance evaluation in real-world settings. This is typically performed using the manually annotated data

New Query : (Mention.class="Person") [Clear] [Execute]

Total Found Patterns : 313 [Export Patterns] [XML] [HTML] [All Patterns] [Selected Patterns]

Annotation Types : Token [Features : All] [Add Annotation Type]

Pattern Text : Conservative vice chairman, said Baroness Thatcher had a "very small"

Token.category : JJ NN NN V... NNP NNP V... RB JJ

Mention.class : Executive Person Woman

Token.orth : upperInitial lo... lowerc... lo... upperl... upperl... lo... lo... low...

Token :

Text : Conservative
Features :
string=Conservative
kind=word
length=12

Document	Pattern	Right Context
ft-extremists-07-oct-2001.xml_0003F	Iain Duncan Smith	on Sunday night signalled his
ft-extremists-07-oct-2001.xml_0003F	Mr Duncan Smith	has turned on the group
ft-extremists-07-oct-2001.xml_0003F	Gary Streeter	, a Conservative vice chairman
ft-extremists-07-oct-2001.xml_0003F	Gary Streeter	, a Conservative vice chairman
ft-extremists-07-oct-2001.xml_0003F	Conservative vice chairman, said Baroness Thatcher	had a "very small"
ft-extremists-07-oct-2001.xml_0003F	should break with her as Tony Blair	had ditched clause 4 -
ft-extremists-07-oct-2001.xml_0003F	commitment to full-scale socialism. Michael Howard	, shadow chancellor, said
ft-extremists-07-oct-2001.xml_0003F	shadow chancellor, said that Mrs Thatcher	had "saved this country was

Figure 3.2: ANNIC corpus analysis tool

from the first phase and running GATE's performance evaluation tools to compare the automatic results against the gold standard. Standard measures such as precision and recall are produced and the application developers can then assess whether the required performance is reached (see [CMBT02] for further details).

With respect to deployment with the end-user application, GATE provides support for connecting to ontology tools such as KAON2 and Jena (see SEKT deliverable [SEK05e] on Integration of GATE with KAON), integration of GATE applications within Tomcat, and as command line tools.

In the following sections we describe some Human Language Technology applications for the Semantic Web which have been specifically designed for use in industrial settings and which have been implemented and tested as part of SEKT. In addition, there are several other success stories and best practices from outside SEKT, which are reported elsewhere [LMG05].

3.2 Automatic annotation

The BT case study makes use of the massive semantic annotation tools developed in SEKT, which encompass ontology-based information extraction, instance disambiguation, and a scalable knowledge store for collecting, reasoning, and accessing the metadata. The documents are analysed automatically in order to populate the ontology and link the documents to the structured knowledge.

It allows the combination of FTS (full-text search, i.e. the simplest form of IR) with structured (DB-like) queries over the semantic metadata. An example of this might be the case of asking for all documents which contain references to objects matching a structured

query and containing a given keyword. The structured queries are performed on top of a semantic store. Based on automated interpretation (reasoning, inference) on top of the semantics of the data (the ontology), the semantic store is capable of answering questions based on data which has not be explicitly asserted. For example, it is capable of returning “Mary” as a result of the query $\langle John, relativeOf, ?x \rangle$ based on an ontology of family relationships and the assertion $\langle Mary, motherOf, John \rangle$.

The SEKT search and browse tools are used to provide user-friendly access to the metadata (see SEKT deliverable [SEK05d]).

3.3 Analysing competency questions

In the SEKT legal case study some 800 competency questions were derived from interviews with experienced judges throughout Spain, as described in section 5.2. Once this knowledge was obtained, the construction of the ontology was based on the term and relation extraction from the questions regarding judicial practical problems posed by the judges during their interviews. Due to the fact that at that time semi-automatic extraction software for Spanish was not available, the extraction is performed manually, nevertheless, tools such as TextToOnto² and ALCESTE were used to support manual term extraction. The SEKT deliverable [CPC⁺04], chapter 2, describes the usage of the tools in great detail, and pins down the experiences and findings.

3.4 Lessons learned

In this chapter we have discussed some examples of semantic web applications using Human Language Technology which have been developed specifically for real use in industry. While at present these are all research systems, they all demonstrate very clearly:

- the need for and importance of such applications in industry;
- the transition between research prototypes and real world applications;
- the actual use of such technologies in industry;
- the performance levels necessary to be of use in an industrial setting.

The applications are still in the process of development and improvement, but rather than being a drawback, this actually serves to emphasise the importance of benchmarking activities and testing applications in the real world, for it is often only through such methods (rather than laboratory testing under ideal conditions and with very basic scenarios) that useful improvements can be made that will benefit end users to the maximum.

²<http://sourceforge.net/projects/textttoonto>

While the results of the best practices questionnaire show us the ideas and opinions about the Semantic Web of those involved in the field, the examples of real applications in use aim to show us in more practical terms which kind of practices are really useful. In particular, it shows that while these applications are not particularly mature indeed, many of them are still ongoing further development and are still in the research phase they are nevertheless useful to real users as they stand. This emphasises the point that often tools and applications can be useful if only semi-automatic or if results are not perfect, because they enable users to save time and money in performing tasks which would previously would have been achieved manually, or with great difficulty by a human.

Another particular point to note is that all these systems are based to some extent on the architecture on GATE, which was designed to be an open and flexible architecture for language processing. GATE has been used in many different ways, not just for information extraction (the application for which it is best known) but as the basis for many different tasks (e.g., summarisation, language generation, ontology population). It has also been designed to work with different languages, scenarios, applications, and domains with the maximum robustness and ease of adaptation. This, we believe, is a crucial point in expanding the prevalence of the Semantic Web.

So, we can summarize that

- in order to enable easy integration of HLT tools within end-user applications, a comprehensive HLT infrastructure is required, which addresses issues such as diverse document formats, multiple languages, scalability, portability, and robustness.
- Multilinguality is particularly important, especially for European companies which supply products to diverse markets. This is especially true for components like lemmatizers and stemmers, which need to be rewritten by experts for other languages as they can't be merely translated.
- given a powerful HLT infrastructure on one hand, and sufficient communication between domain experts and application builders (e.g., ontology engineers, HLT experts), existing tools or new tools can be built or adapted for specific needs with relatively little effort.
- adhering to a proper development cycle for HLT-based applications is crucial: starting from data collection, component modification, and finally quantitative evaluation on the data to determine how effective the HLT technology is in addressing the user requirements.

Chapter 4

Ontology Management Technologies

Next Generation Knowledge Management solutions will be built upon ontology-based metadata and thus the creation and management of machine-interpretable information and the consequent use of ontologies. The integrated management of ontologies and metadata, especially their generation, mediation and evolution, is fundamental to this development and relies in part on innovative Human Language Technologies and Knowledge Discovery methods.

4.1 Basing an Ontology on a Foundational Ontology

This section describes the lessons learned from the integration of PROTON¹, an upper-level ontology, and of the Ontology of Professional Judicial Knowledge (OPJK). More information on both ontologies may be found in the respective deliverables.

4.1.1 The Structure of PROTON

In order to meet the requirements of the usage scenarios and to assure easy and gradual understanding, PROTON is separated into four modules:

- **System module.** It contains a few meta-level primitives (5 classes and 5 properties). It introduces the notion of 'entity', which can have aliases. This module can be considered an application ontology. Within this document and in general, the System module of PROTON is referred to via the `protons:` prefix.
- **Top module.** The highest, most general, conceptual level, consisting of about 20 classes. These ensure a good balance of utility, domain independence, and ease of understanding and usage. The top layer is usually the best level to establish

¹<http://proton.semanticweb.org>

alignment to other ontologies and schemata. Within this document and in general, the Top module of PROTON is referred to via the `proton:top` prefix.

- **Upper module.** Over 200 general classes of entities, which often appear in multiple domains. Within this document and in general, the Upper module of PROTON is referred to via the `protonu:` prefix.
- **KM (Knowledge Management) module.** 38 classes of slightly specialized entities that are specific for typical KM tasks and applications. The KM module is actually the former SKULO ontology, further developed and integrated into PROTON. Within this document and in general, the PROTON KM module is referred to via the `protonkm:` prefix.

The design at the highest level (immediately below the Entity class) of the Top module makes the most important philosophical and modelling distinctions. It follows the stratification principles of DOLCE [GGM⁺02] through the establishment of the PROTON trichotomy of Objects (`dolce:Endurant`), Happenings (`dolce:Perdurant`), and Abstracts (`dolce:Abstract`). A detailed documentation of the ontology, including its link to other ontologies and metadata standards can be found in [TKM04]. PROTON is available online².

4.1.2 OPJK Integration

It is important to note that the knowledge modelled corresponds to the professional (not the purely theoretical) knowledge contained within the judiciary and transmitted only between its members [CCP⁺05]. We have described this fact more deeply in section 5.2 (see also [CBK⁺05]).

In order to gain an upper conceptual framework, we integrated PROTON and OPJK. Using PROTON, we not only reuse an existing upper ontology but also are able to maintain the necessary professional trait of OPJK. This integration has taken place at two stages: OPJK concept into PROTON concept integration and the reuse of existing PROTON relations.

4.1.3 Class Integration

The first part of the integration process consisted mainly in generalizing OPJK concepts taking fully into account the System and Top modules of PROTON. The top layer of PROTON is the best level to establish the alignment with OPJK. It has proved to be domain independent and its easy understanding and usage have been essential to guarantee this integration. The primary classes

²<http://proton.semanticweb.org>

Abstract, Happening and Object were straightforwardly incorporated, although Abstract needed the introduction of a specific subclass `AbstracionLegal` [`LegalAbstraction`] for organizational purposes. With this domain specific consideration, the OPJK classes `CalificacionJuridica` [`LegalType`], `Jurisdiccion` [`Jurisdiction`] and `Sancion` [`Sanction`] could be better related and specific relations between them, not shared by the rest of classes or instances within Abstract.

The class of entities Happening, which includes Event, Situation, and TimeInterval, is able to directly incorporate the fundamental OPJK classes `Acto` [`Act`], `ActoJuridico` [`JudicialAct`], `Fase` [`Phase`] and `Proceso` [`Process`]. These classes contain the taxonomies and relations related to all legal acts, to the different types of judicial procedures (subdivided within civil and criminal judicial procedures) and the different stages that these procedures can consist in (period of proof, conclusions). We need to stress the introduction of the class `Role` to PROTON, which allowed the distinction of situations where an Agent (Organization or Person) might play a part in situations that differ from the more business-related `JobPosition`. In the case of OPJK, the class `Role` contains the concepts and instances of procedural roles (`RolProcesal`) that an agent might play during a judicial procedure.

Finally, Object includes the top OPJK classes `Agent` and `Statement` that are generalizations for `Document`, and `Location`, necessary concepts to contain, within others, the organizational taxonomy of courts (`OrganoJudicial`), and judicial documents (`Contrato` [`Contract`], `Recurso` [`Appeal`], `Sentencia` [`Judgment`], etc.).

4.1.4 Inherited Relations

The specificity of the legal (professional) domain requires specific relations between concepts (normally domain-related concepts as well). However, most existing relations between the Top module classes taken from PROTON have been inherited and incorporated. It has not been necessary for the usage of the *Luriservice* prototype to inherit all PROTON relations, although most of the relations contained in PROTON had already been identified as relations between OPJK concepts.

The following relations – not a comprehensive list – have been inherited from the existing relations in within the Top module concepts: `Entity` `hasLocation`, `Happening` `has endTime` and `startTime`, `Agent` `is involvedIn` (`Happening`), `Group` `hasMember`, an `Organization` `has parent/childOrganizationOf` (`Organization`) and `is establishedIn`, and, finally, `Statement` `is statedBy` (`Agent`), `validFrom` and `validUntil`.

4.1.5 Lessons learned

The work described here shows that modularisation and the light-weight approach allowed the Top module of PROTON to meet a good balance between utility and ease of use and understanding. As top layers represent usually the best level to establish alignment to other ontologies, the classes contained in the Top module *Abstract*, *Happening* and *Object* were straightforwardly incorporated, together with most of their subclasses. In the same way, most of the existing relations between the Top module classes were also inherited. Had PROTON have been a more monolithic ontology, the integration would be seriously hampered by the sheer size of the two ontologies. Reusing just the top module enabled us to finish in a well defined time frame and with comprehensible results.

The essential domain independence of PROTON has fostered this integration and both, the reuse of existing PROTON classes and relations and the creation of new specific legal domain subclasses or relations to facilitate OPJK integration illustrates the flexibility of this domain independent top ontology.

For the integration, several concepts (like *AbstracionLegal* or *RolProcesal*) were introduced solely to cluster other concepts by topic. Although there is no ontological justification for this, this is a powerful tool to ease the engineering and understanding of ontologies, increasing their sharedness, because concepts that belong together will be shown together.

We summary the lessons learned from the integration:

- reusing an upper level ontology can ease the process of creating one from scratch – if they indeed are easily reusable (4.1).
- making modules instead of one big monolithic ontology enables reuse more easily.
- light-weight ontologies make for an easier reuse by non-experts, as they don't have to commit to logical consequences they maybe don't realize. This also lends to an easier understanding than heavy weight ontologies provide.
- simple names, labels and sufficient comments of ontological entities are important. It is easier to reuse *Happening* and *Object* than *Endurant* and *Perdurant*, respectively, especially when they are well-documented.
- a well-defined process to feed back change requests lead to less frustration on the (re)users' side.
- clustering ontological entities by topic under a common concept eases up the engineering and understanding of an ontology.

4.2 Managing Ontologies

If we compare ontologies with the schemata of relational DBMS', there is no doubt that the former represent (or allow for representations of) richer models of the world. There is also no doubt that the interpretation of data with respect to the fragments of first order logic, which are typically used in knowledge representation (KR), is usually computationally more complex than the interpretation of a model based on relational algebra. From this perspective, the usage of the lightest possible KR approach, i.e. the least expressive logical fragment, is critical for the applicability of ontologies in a bigger fraction of the contexts in which DBMS are used, due to the available tool support.

From our point of view, it is very important for the DBMS paradigm that it allows for the management of huge amounts of data in a predictable and verifiable fashion. This requirement is not well covered by heavy-weight, full-fledged, logically expressive knowledge engineering approaches. Even taking a trained knowledge engineer and a relatively simple logical fragment (e.g. OWL DL), it becomes hard for the engineer to maintain and manage the ontology and the data, with the growth of the size of the ontology and the scale of the data.

So we restricted ourselves in several way with regards to the used KR language, OWL [SWM04]. First, PROTON follows the principle of strict layering, which means that no classes or properties are instances of other classes. This restriction is already introduced in the OWL DL sublanguage. We also used only a certain part of the available language constructs, basically OWL DLP [Vol04]. OWL DLP can be used in a running system in a very efficient manner, as it allows to fully materialize all logical consequences of the knowledge base and thus lends to fast responses. This means that instead of inferring answers on the fly when the question is asked, all possible answers are explicitly saved before being questioned. This needs more space, but leads to faster response times. OWLIM [KOM05], developed by OntoText within the SEKT project, implements OWL DLP, in a fast and easy to manage manner.

4.2.1 Lessons learned

With limiting ourselves to the above modelling means and principles we achieved to reap a number of advantages: (i) this level of expressivity allows for straightforward reasoning and interpretation on the basis of standard extensional semantic; (ii) it matches the modelling paradigm of the object-oriented approach (of OO-databases, or languages like Java); (iii) it allows easy maintenance, integration and migration to other formalizations.

It is unclear, if the problems of complex managing were inherent due to the expressive power of the used language (i.e. OWL), or if it is due to the lack of proper tool support. Better tool support, especially as developed within the SEKT project and in close corporation with the affected case study partners, is currently being assessed.

To summarize:

- in order to manage ontologies more easily, it often can be fruitful not to exploit their whole expressive power, but only as much as needed for the task at hand.
- as we can see in the case studies, even “a little semantics can go a long way”. We don’t have to use the full expressive power of a language like OWL DL in order to benefit from ontologies.

Chapter 5

Methodology

In this chapter we will take a look at the SEKT methodology, called DILIGENT, and how it was applied in the SEKT case studies until now. We will identify the lessons learned when using DILIGENT.

We will see, that different parts of DILIGENT suit best the specifics of each case study. We will identify these parts and interpret the work done so far through the DILIGENT framework view. We benefitted threefold from applying an explicit ontology development process like DILIGENT to the case studies:

1. specific problems of the case studies are addressed and attacked with DILIGENT, for example in the legal case study, where the argumentation framework has been applied and the discussion focussed and shortened while creating the the legal ontology,
2. using a framework like DILIGENT helps in describing the ontology lifecycle within the case studies in a more concise and generic way, in order to gain accessibility to the experiences won in the case studies, and
3. for the refinement of DILIGENT, we have been able to harvest experience with using the process and getting feedback in order to identify problems and describe the steps of DILIGENT in greater detail.

Besides the case studies, the development of an upper level ontology for SEKT became a further interesting application of the DILIGENT process. In the following section we will first give a short description of DILIGENT. Then we continue with highlighting several lessons learned that came up during the application of DILIGENT in the case studies. We summarize the lessons learned at the end of each section.

5.1 Overview of the SEKT methodology

In order for this report to be self-contained, we offer here a short description of the SEKT methodology. The methodology is named DILIGENT, which stands for the *DI*stributed, *Loosely controlled and evolvInG Engineering of oNTologies*. A detailed and current description of DILIGENT is found in [STV⁺05].

As shown in Figure 5.1, the DILIGENT process comprises five main activities: (1) **build**, (2) **local adaptation**, (3) **analysis**, (4) **revision**, (5) **local update**. The process starts by having *domain experts, users, knowledge engineers* and *ontology engineers* **building** an initial ontology. In contrast to known ontology engineering methodologies available in the literature [GPS98, GPFLC03, PM01, UK95] our focus lies on distributed ontology development involving different stakeholders, who have different purposes and needs and who usually are not at the same location. Therefore, they require online ontology engineering support.

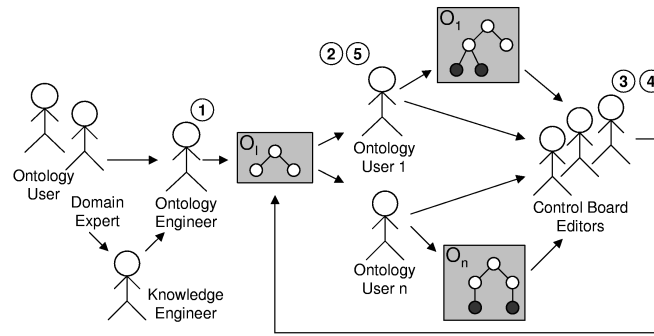


Figure 5.1: Roles and functions in distributed ontology engineering

The team involved in building the initial ontology should be relatively small, in order to more easily find a small and consensual first version of the shared ontology. Moreover, we do not require completeness of the initial shared ontology with respect to the domain.

Once the product is made available, users can start using it and **locally adapting** it for their own purposes. Typically, due to new business requirements, or user and organization changes, their local ontologies evolve [KF01] in a similar way as folder hierarchies in a file system. In their local environment they are free to change the reused shared ontology. However, they are not allowed to directly change the ontology shared by all users. Furthermore, the control board collects change requests to the shared ontology.

The board **analyses** the local ontologies and the requests and tries to identify similarities in users' ontologies. Since not all of the changes introduced or requested by the users will be introduced,¹ a crucial activity of the board is deciding which changes are going to

¹The idea in this kind of development is not to merge all user ontologies.

be introduced in the next version of the shared ontology. The input from users provides the necessary arguments to underline change requests. A balanced decision that takes into account the different needs of the users and meets user's evolving requirements² has to be found. The board should regularly **revise** the shared ontology, so that local ontologies do not diverge too far from the shared ontology. Therefore, the board should have a well-balanced and representative participation of the different kinds of participants involved in the process.

In this case, users are involved in ontology development, at least through their requests and re-occurring improvements and by evaluating it, mostly from an usability point of view. Knowledge providers in the board are responsible for evaluating the ontology, mostly from a technical and domain point of view. Ontology engineers are one of the major players in the analysis of arguments and in balancing them from a technical point of view. Another possible task for the controlling board, that may not always be a requirement, is to assure some compatibility with previous versions. Revision can be regarded as a kind of ontology development guided by a carefully balanced subset of evolving user driven requirements. Ontology engineers are responsible for updating the ontology, based on the decisions of the board. Revision of the shared ontology entails its evolution.

Once a new version of the shared ontology is released, users can **update** their own **local** ontologies to better use the knowledge represented in the new version. Even if the differences are small, users may rather reuse *e.g.* the new concepts instead of using their previously locally defined concepts that correspond to the new concepts represented in the new version.

5.2 Social Issues of Ontology Engineering

Gathering the data an ontology is based on is not just a technical issue. As we will outline in this section, it is also and especially a social issue, and much care must be put into understanding the people we are aiming at and their vocabulary and point of view on their environment. The SEKT legal case study gained rich experiences with regards to this problem.

The goal of the legal case study is to provide support to judges in their first years. In the Spanish system one particular problem young judges face is when they are on duty and faced with situations in which they are not sure what to do (*e.g.* in a voluntary confession, which process of questioning should be applied?). In such cases, they usually phone their former training tutor (experienced judges) for resolving the issue. But this is a slow and insecure procedure. In this case study, an intelligent system is being developed to speed up the process and to relieve experienced judges from this effort by providing support to young judges. Only in the case that the requested knowledge is not in the system

²This is actually one of the trends in modern software engineering methodologies (cf. Rational Unified Process [BR96]).

and cannot be reformulated from already stored knowledge, an experienced judge will be contacted. The result of this “expensive” consultation will be fed back into the system automatically.

In order to build a scalable and useful system, several requirements have been identified [RCCP04]. In particular, the decision as to whether a request for knowledge is covered by the knowledge stored in the system should be based on semantics of the legal domain rather than on simple word matching. An ontology can be used to perform this semantic matching. Case-based reasoning techniques will be used when considered applicable. The main difference with traditional CBR approaches is that we will use a semantic-based similarity measure based on ontologies. Moreover, cases will be - where possible - automatically extracted from information generated by judges as they perform their daily work.

5.2.1 Capturing Professional Judicial Knowledge

Professional Legal Knowledge refers to the core of professional work that contains the experience of the daily treatment of cases and is unevenly distributed within individuals as a result of their professional and personal experiences. The knowledge acquisition process has been based on an ethnographic process designed by the UAB team and the Spanish School of the Judiciary within the national SEC project, to efficiently obtain useful and representative information from questionnaire-based interviews. Nearly 800 competency questions have been extracted from these interviews and the ontology is being modelled from the selection of relevant terms.

So before we started to model the ontology, we performed social knowledge acquisition. Capturing professional knowledge is a time consuming and often painstaking process implying different types of social techniques (usually surveys, interviews, participant observation, focus groups and expert panels). This means inferring social knowledge from protocols. The way in which this set of tasks is performed usually influences the ontological modelling. This problem deserves a separate reflection on what we will call “pragmatic integrated cycle” (from knowledge acquisition and ontology construction to the users’ validation plan).

Ontologies of Professional Legal Knowledge would model the situated knowledge of professionals at work. In our particular case we have before us a particular subset of Professional Legal Knowledge belonging specifically to the judicial field. Therefore, we will term the conceptual specification of knowledge contained in our empirical data Ontology of Judicial Professional Knowledge (OPJK). Modelling this professional judicial knowledge demands the description of this knowledge as it is perceived by the judge and the attunement of dogmatic legal categorizations. The way in which judges produce a different kind of knowledge through dogmatic legal categorizations is not clear yet. But the assumption that their reasoning process follow some specific dogmatic patterns is not required.

To model this ontology, first, we have had to acquire the judicial professional knowledge as it can be collected and reconstructed from regular data. The work on the ethnographic field offered us a set of protocols (literal transcriptions of the interviews, the completed questionnaires and the extracted questions) containing this knowledge.

Collecting this competency questions was hard. Judges use a different vocabulary, and usually don't offer the interviewer simple competency questions but rather reflect on problems they had narratively. Competency questions had to be formulated out of this narrations, and still retaining the legal language the judges used, as it is probable that other judges will use this language as well when interacting with the system.

Once the knowledge is obtained, the construction of the ontology is based on the term and relation extraction from the questions regarding judicial practical problems posed by the judges during their interviews. Only the competency questions were taken into account for building the ontology. If someone was extending the ontology with concepts based on legal theories, they were removed. We learned that legal professional knowledge as used by judges in their daily routine does not follow the dogmatic and normative legal theories.

5.2.2 Lessons learned

Although there have been numerous legal ontologies, a careful evaluation of the requirements of the SEKT legal case studies showed that even though they all modelled the legal domain, they modelled theoretical legal knowledge, legal theory. That type of modelling was inappropriate for the task at hand. In the legal case study the knowledge to be modelled was practical, based on professional issues of judges, and did not need to take legal theory into account to model this practical domain.

An extensive survey [CCV⁺05, PC05] was done to understand these judges and to gather more than 800 competency questions. This bases the OPJK on solid ground with regards to the requirements of the users of the SEKT legal case system. As we will see in chapter 2 the competency questions were also used to evaluate the ontology.

OPJK modelling affirms that the modelling of professional judicial knowledge demands the description of this knowledge as it is perceived by the judge and the abandonment - or at least the attunement – of dogmatic legal categorizations.

The usual Knowledge Acquisition literature, written by engineers, was perceived by the legal team to be insufficient. They often lack the description of the kind of problems the team encountered when dealing with judges and expliciting their knowledge.

To summarize, we identify the following advices:

- certain types of ontologies, especially those related to professional knowledge, have to be build by domain experts and ontology engineers together. So it is important that they have tools (distributed environment) to communicate changes in the on-

tology. And as domain experts have to be involved in the process, tools have to be domain expert friendly – not just for nerds.

- competency questions should be collected from the actual users, and not created by knowledge engineers or theoreticians.
- sometimes it is hard to speak to the expected users, because they may have a different background, live in a different culture, basically speak a different language. But even though it is hard – it is worthwhile.
- professional knowledge is different than theoretical knowledge. When building ontologies that are to model professional or tacit knowledge, stick to the collected competency questions and ignore other sources.

5.3 Using off the shelf tools

5.3.1 Ontology engineering in a wiki

In order to implement the DILIGENT methodology, we first went for easy to get tool support with existing off the shelf tools. For capturing the argumentation about an ontology, a wiki [CL01] was suggested. The wiki was the tool of choice because of the ease of use the technology promises and due to the availability of implementations of the technology. Discussing concepts was considered to be very easy with the help of the wiki. The success of projects like Wikipedia was taken as an indicator towards the successful use of the technology. A wiki also offers an accessible web based interface, thus allowing the SEKT team, being spread over Europe, to access the current version of an ontology. This allows external users to track the discussion.

For the BT Digital Library case study it was decided to build an ontology based on the PROTON ontology [TKM04]. Some changes and extensions had to be made to the ontology in order to make it fit the needs of the Digital Library better. These changes were to be tracked and discussed in a wiki.

Using the wiki was (initially) more cumbersome than using email for discussing the development of the initial BT Digital Library ontology. This was mainly due to the convenience of using email; most people responded or contributed pretty quickly to 'interesting' discussions. Discussion on the wiki was very limited as people could not get into the 'culture' of accessing the wiki, without being prompted to do so, and then discussing or commenting. An 'alerting' mechanism would have helped here, which was not part of the used wiki implementation. So the main problems were really with the implementation. Furthermore, the initial BT ontology used many of the classes and properties from PROTON; very few new classes and properties needed to be defined. The discussion (via email) was therefore brief.

5.3.2 Lack of tool integration

At the same time the engineering of the ontology for the Legal Case study was going on. As the ontology was to be bigger and more independent from the existing PROTON ontology, we thought it would benefit more from the use of the wiki. And the wiki did help a lot: it made the ontology discussion transparent, traceable and available to all members of the team.

However, the tool did not provide several important features such as: visualization of the graphical representation of the ontology being built or a system of e-mail notifications when arguments had been added. To solve the requirement of graphical visualization, the ontology modelling team extended the wiki with screenshots from the relevant parts of the ontology build with KAON Oi-Modeller [GSV04]: they modelled the ontologies agreed on in the KAON Oi-Modeller, made a snapshot of part of the ontology and imported it to the wiki, in order to visualize the ontology and ease the discussion and understanding of it. In figure 5.2 we see a screenshot of the wiki, running on the SEKT portal, showing the concept *Hecho* with its description, argumentation and attributes, as well as a graph made with the KAON Oi-Modeller showing the concept and its attributes.

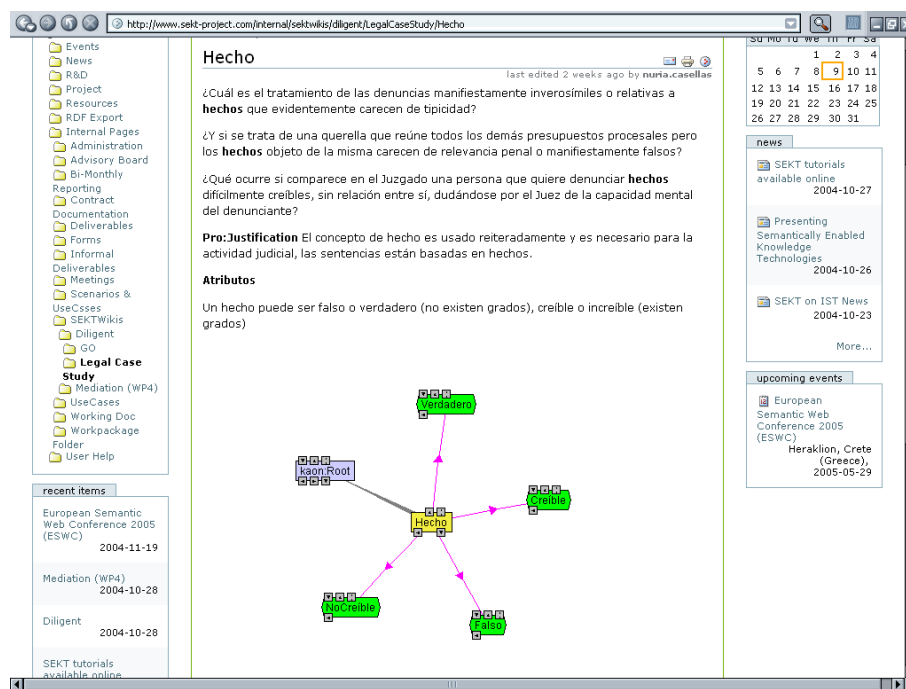


Figure 5.2: Screenshot of the wiki used for discussing the legal ontology

As we have seen the wiki technology allowed for a much better tracking of the argumentation than the previous approach. The effectively used engineering system was made up of several tools, used in parallel (thus leading often to work done more than once, due to the lack of interoperability of the tools [SA02]). These tools were the wiki, used for

the tracking of the argumentation; the KAON Oi-Modeller, used for the visualisation of the ontology; and Protégé, used for the formalisation of the ontology.

5.3.3 Ontology maintenance in a forum

The PROTON ontology is an upper-level ontology developed within the SEKT project [TKM04]. It contains about 300 classes and 100 properties, providing coverage of the general concepts necessary for a wide range of tasks, including semantic annotation, indexing, and retrieval.

In order to set up a dedicated community process to handle change requests and to capture comments on PROTON we were setting up a wiki in combination with a forum. The use case was different from the other case studies: instead of creating a new ontology we had to deal with a grown and quite big ontology.

The wiki should hold the description of all concepts and relations of PROTON, whereas the forum should host the discussion about them. Wikis are great as easy to use and edit web-based content management systems, and thus lend perfectly to the given task. Forums on the other hand have proofed to handle discussions for non-technical groups for years and enjoy a high popularity among web users. We hoped to keep discussions alive more easy with this setting and separation of concern.

Creating the wiki pages for all concepts and relations of PROTON would have taken a long time. We knew we had to automate this step. The implementation of the system was done in close cooperation between OntoText and the AIFB. The wiki we used was based on Zope³, a Python⁴ based web content management system and also the base for the AIFB Semantic Portal [MSS⁺03]. As Zope is easy extensible with further Python scripts, we created scripts to import an ontology into the wiki and create a first documentation based on the labels and comments in the ontology. The second step was installing the forum, and finally connecting the wiki and the forum. So discussions would be performed in the forum, and the results would be rendered in the wiki.

Work continues on integrating tools to further the discussion on PROTON, as described in [sek].

5.3.4 Lessons learned

We have applied existing tools to quickly create environments where DILIGENT could be applied for the ontology lifecycle. Even though the tools had some considerable drawbacks, we could validate the ideas behind the used tools: as we will see in the next section, tracking ideas has proofed as considerably speeding up the development of ontologies and allowing the distributed engineering of ontologies. It also was great for the dissemination

³<http://www.zope.org>

⁴<http://www.python.org>

about the current state of an ontology, as the wiki was a web site, accessible from everywhere.

Even though email has drawbacks – like a closed group of recipients or the difficulties in browsing an email archive later on – this well-known technology was favoured by users in the BT case study, because lack of proper notification facilities in the wiki were not allowing for a interactive discussion.

The highly motivated team of legal experts building the OPJK extended the tool to meet their requirements for an ontology argumentation environment. Even though it meant a lot of additional manual effort, they extended the wiki with externally created visualizations and used the wiki for both documentation and argumentation. They had several versions of the OPJK, all of them synchronized and extended manually. This was especially due to the ontology tools not interoperating among each other properly. If they would work together, we imagine even non-technical research groups to be able to solve a much wider range of problems in a creative way than they could now.

The experiences with the applied tools lead to the following lessons learned:

- off the shelf tools help in the first validation of research ideas, but have to be replaced by dedicated tools for real application.
- a strong integration between ontology tools is crucial. This will help users to come up much more easily with new and innovative ideas to use given tools.
- visualisation is crucial. The users have gone great lengths to provide a visualisation, even if it meant a lot of manual and repetitive work.

Based on the experiences in the case studies, the design of a DILIGENT tool was created in close cooperation of BT and the AIFB. This design is presented in chapter 6.

5.4 Faster Ontology Engineering

5.4.1 Focussing the discussion

The method used to build the OPJK ontology has focused on the discussion within the UAB legal experts team over the terms which appear on the competency questions. It starts with the selection (underlining) of all nouns (usually concepts) and adjectives (usually properties) contained in the competency questions. Once the terms had been identified, the team discusses the need to represent them within the ontology and their place within the taxonomy.

Soon they realized that they were often discussing questions at length before making a decision. But after a few weeks, the decision was not traceable any more, and they

started discussing the original question anew. This first phase of ontology development took about half a year, but yielded not many tangible results in forms of agreed ontological entities. This time was also a very important learning time for the legal team. It included a lot of reading on ontology engineering.

In a joint workshop of the legal team and the AIFB ontology team DILIGENT was proposed and the legal experts were trained in applying a methodology to the ontology engineering process. The argumentation framework of DILIGENT was identified as being able to help solving problems such as missing traceability and explicitness. For that reason, the introduction of the Distributed, Loosely controlled and evolving Engineering of oNTologies (DILIGENT [TPSS05]), provided by the AIFB research team, offered a reliable basis for a controlled discussion of the arguments for and against a modelling decision. The introduction of DILIGENT proved the need to rely on guidelines for the decision-making process within ontology design. The use of DILIGENT speeded up the modelling process, as decisions were more easily reached and more concepts were agreed upon.

DILIGENT requires the discussion to be traceable and to make the arguments used in favour or against the introduction of a certain concept explicit. The visualization of the arguments takes place on a wiki-based environment.

From our point of view argumentation visualization is mature from the research perspective. First attempts were made to combine findings from argumentation visualization and ontology engineering. However, as it is argued in [PB88, dMA03] argumentation is best supported when the methodology such as IBIS is customized with respect to the domain which is argued about. Hence, research is moving into the following directions:

- Identify the most relevant arguments in ontological discussions.
- Support synchronous as well as asynchronous discussions.

5.4.2 Experiences from ontology building

In the next half a year, the legal team proceeded by moving from one competency question to the next, ignoring outside sources (like theoretical works) as not relevant to the discussion at hand. They managed to integrate 200 competency questions and create an ontology with about 700 ontological entities, i.e. relations, concepts and instances.

It is interesting to note that the discussion centered around the relations. Concepts and instances were relatively easy to agree on, but the meaning and the role of many relations were a hotly debated topic. This was an important insight for designing tools to help in the shared engineering of and argumentations on an ontology.

Another result, although expected, was that the number of new entities was very high with the first competency questions, but got considerably smaller with further questions. At the same time, discussions took longer and surfaced more often. This could either

indicate that there is a sweet spot for rapid ontology development, or that the first entities were not specified enough to identify their conceptualizations properly and thus turned open to debate. As DILIGENT does not require the first version of the released ontology to be complete, this indicates towards the rapid development DILIGENT could be used in.

The UAB team progressed by having a small number of very intensive and long meetings (8 meetings within 1-2 weeks, each about 10 hours long). They are convinced that a less intensive work would have prolonged the engineering time of the OPJK considerably, because it always needed a long time to enter the discussions at the required depth. They discussed almost solely content questions: technical issues of how to formalize agreed on conceptualizations and how to express certain ideas with the used ontology representation language were negligible. Maybe this is due to the formal specification being done by only one team member after the discussions (performed in Protégé [Noy02]). Questions about the ontology language were resolved in direct conversations with ontology experts.

5.4.3 Capturing the argumentation

There is evidence that distributed ontology development can be rather time consuming, complex and difficult, in particular getting agreement among domain experts. Therefore, one needs an appropriate framework to assure it in a speedier and easier way. In order to provide better support, we identify the kind of arguments that are more relevant and effective to reach consensus and restrict the discussion accordingly. The Rhetorical Structure Theory (RST) [MT87] was applied to classify the kinds of arguments most often used and identify the most effective ones.

Previous experiments performed [PSST04] provide strong indication—though not yet full-fledged evidence—that a restriction of possible arguments can enhance the ontology engineering effort in a distributed environment. Moreover, the middle-out strategy combined with appropriate argumentation support and management technologies can be used to quickly find a shared, consensual ontology even when participants must provide all and only written arguments.

A central issue in the DILIGENT process is keeping track of threads of exchanged arguments. We can identify several stages in which arguments play an essential part:

- Ontology is defined as “a shared specification of a conceptualization” [Gru95]. Although “shared” is an essential feature, it is often neglected. In DILIGENT experts exchange arguments while building the initial shared ontology in order to reach consensus;
- When users make comments and suggestions to the control board, based on their local adaptations, they are requested to provide the arguments supporting them;

- While the control board analyses the changes introduced and requested by users, and balances the different possibilities, arguments are exchanged and balanced to decide how the shared ontology should change.

For capturing the argumentation a wiki [CL01] was used, as described in the previous section.

Keeping the argumentation stack – a kind of to-do list of open issues – up to date and discussing concepts was very easy with the help of the wiki. Besides the argumentation stack, an alternatives stack would be helpful. Argument types, in particular *elaboration*, *evaluation* and *justification* and *alternatives*, were used heavily during the experiments. However, the lack of appropriate evaluation measures made it difficult, at some times, for the contradicting opinions to achieve an agreement. The argumentation should then be focused on the evaluation criteria. The evaluation can take place off-line, or can be based on modelling advices from practical experience.

5.4.4 Lessons learned

We have seen how the legal experts were first overwhelmed with the task of creating the ontology. Discussions were going on, and often repeated. There was a lack of direction and goal-orientation while discussing the concepts, which was solved by introducing the DILIGENT methodology. Afterwards the experts were more aware of the ontology engineering process and could estimate their progress in a much clearer way.

Besides the general advice of applying a methodology, the argumentation framework of DILIGENT proved to be valuable for the focussed work on the ontology. Restraining themselves to relevant arguments and especially recording the arguments in a traceable and comprehensible way helped rendered ontology engineering process much more efficient.

To track the arguments and direct the discussion, the introduction of the role of a moderator in the ontology development team was useful. Thus we can ensure both the more complete capture of the argumentation, and a stronger focus of the discussion.

The development of the OPJK after the introduction of DILIGENT and the argumentation wiki sped up considerably. Other team members felt much better informed and were able to keep up with the discussion even when they were geographically separated.

Still, argumentations sometimes could not be resolved. A set of applicable and easy to follow ontology evaluation criteria would have helped in the discussion, but were only marginally available. We consider this a still open research issue and will follow it in the next year to come up with evaluation criteria for ontologies helpful in the creation and evolution of ontologies.

In summary:

- it is important to follow a methodology when building an ontology – it both clarifies the objectives and provides a list of things to do.
- when building an ontology, tracing the argumentation speeds it up considerably.
- it is important to find good evaluation measures for the ontology once build.

Part II

Tools

Chapter 6

EvA tool design

Based on the various experiences with ontology engineering and the usage of DILIGENT in SEKT, we have identified proper tool support to be crucial. First experiments using off the shelf software like wikis and a forum taught us valuable experiences, as described in section 5.3. In close cooperation between BT and the AIFB, incorporating the experiences from all case studies and the PROTON development, and based on the DILIGENT methodology the following application design was created.

This application design guide described in this chapter focuses on enabling distributed ontology engineering. Distributed in the sense of the here described application means, that the participants of the engineering process are geographically separated. The application follows a client-server approach. A central ontology engineering server is used to store the ontology persistently. The (rich) client allows the manipulation of the ontology by users. All changes in the status of the ontology are propagated as soon as possible (if possible in real time) to all clients. For ensuring an efficient ontology engineering process, the DILIGENT process is employed [STV⁺05]. For focusing the argumentation within the tool the DILIGENT argumentation framework is used [TPSS05], guiding discussions among domain experts and ontology engineers.

EvA, the application described here, is especially suited for the engineering of and argumentation about the TBox of an ontology, also called its vocabulary or schema. It is not meant to replace or act as a full fledged ontology editor. The visualization will not support a huge number of entities as they may generally appear. But usually the TBox remains a small size. Also the discussion about the TBox usually involves a much smaller group. Even though there is no limit towards the number of team members participating in the scenarios described in section 6.3, we may assume that only a small- or middlesized team will allow for the efficient usage of the tool. In chapter 7 on the other hand we will describe a tool that is suitable for the flexible population of ontologies, i.e. the editing of its so called ABox. As this is a process which is potentially followed by a much wider audience, we base the tool in chapter 7 on the MediaWiki software, that has already been proven to work for a very large and heterogeneous audience.

The chapter gives a short overview about the application functionality, and then moves on to describe the personas and scenarios used within the design process. The usability goals and platform constraints are described, followed by a brief process description of the work model. The conceptual design is laid out in detail in appendix A. A glossary (appendix B) offers a definition of the relevant terms and how they are used within this design. We close with the evaluation, a list of open issues and a first glance at the upcoming implementation of the tool.

The methodology followed to create the design (using personas and scenarios) is described in [CR03].

6.1 Overview of Application's Functionality

The application being described here is an ontology editor, which focuses on employing the DILIGENT ontology engineering process including its argumentation model. The main functionalities of an ontology editor such as creating and modifying concepts, relations, and instances are unchanged. Mainly functions for organizing and tracking requirements (so called *sources*), and functionality for arguing over different ways how to implement an ontology are introduced. Additionally, task management functionality is also designed as it can be found in existing ontology editors.

The management of sources allows importing short text snippets, which can hold requirements in the form of competency questions, or different kinds of resources. Technically these resources are all URIs.

The ontology editor design mostly focuses on integrating the DILIGENT argumentation framework tightly into an ontology editor. The DILIGENT argumentation framework uses the following main concepts: *issues*, *ideas* and *arguments*. An *issue* is raised to solve a problem or fulfil a requirement. It is just a high-level description of what should be done in terms of creating, moving, changing, and deleting preliminary modelling primitives. After an issue has been raised, *ideas* are created. Ideas are preliminary ontology change operations. Ideas are expressed by manipulating modelling primitives. These modelling primitives are not yet part of the ontology. Ideas only change the ontology when either the engineering team has voted in favour of them or the ontology owner has accepted them. Contradicting ideas can be expressed by creating new idea sets. These allow the creation of a differing ontology design by hiding and changing as well as adding modelling primitives. Before voting on an ontology design choice different types of *arguments* can be exchanged by the participants of the ontology engineering process. Generally only pro and contra arguments and examples and counter examples are valid arguments to be used by participants. An automatic classification of the argument is still an open research issue. For now the classification of arguments needs to be done manually by the participants.

6.2 Personas

6.2.1 Persona Justine

Justine is working for a research organization, which focuses on exploring ways on how to use information technology for helping solicitors do their job. Justine believes that solicitors should make use of modern technology to devote more time to their actual task of establishing justice. She is in her early 30's and ambitious about achieving goals in her work environment. She is travelling a lot to meet people at different conferences. One important research topic for her and her research group is to make practical and theoretical knowledge easily accessible for solicitors. Justine's background is in law and she is now pursuing her PhD. Therefore Justine is a domain expert in her field.

Justine has exclusively used the Microsoft operating system Windows (by now she uses Windows XP). She has in depth skills of Microsoft Word.

Justine had no knowledge of ontologies until she started to work on her PhD. She read a couple of introductory articles and manuals, but she has no in depth knowledge of ontologies. She also read an introduction to the DILIGENT methodology and understands the DILIGENT argumentation framework.

Justine is interested in engineering an ontology for the research organization she is working at. She has little time and patience to train herself a new software. The best for her would be to have an application that truly supports engineering the ontology and is easy to learn. She understands ontology engineering to be very time consuming because discussions can become rather philosophical.

The technical part of engineering an ontology should be done with the least mental effort possible for Justine. This will help Justine to fulfil her work goals. As Justine sometimes has to review other peoples work and as she has to interrupt her work on the ontology for a couple of days, it is critical for her that the application provides her with tools to enable a quick understanding of the ontology at hand.

6.2.2 Notes on Persona Jack

The persona Jack is playing a minor role in the scenarios. Jack is a system administrator and has not much interaction with the software. He is just responsible for keeping the systems running.

6.2.3 Notes on Persona Richard

Richard is an ontology engineer. He is not trained in law, but has a computer science degree. He is knowledgeable in creating formal correct ontologies. He also plays a minor role in the scenarios. For the scenario "Creating an ontology with the help of software

agents” he plays a more important role because he is the one who will configure the software agent for ontology learning. A clear description of the persona is out of scope because the scenario “Creating an ontology with the help of software agents” is only summarized.

6.3 Scenarios

In the following scenarios are developed, which describe in a practical sense what functionalities of the applications are important to users. The scenario “Learning the application” is introduced to ease the way of learning the new application. The scenarios “Setting up a new Ontology Development Project” and “Arguing over an ontology in an online-setting” describe the core functionalities of the application. The scenario “Arguing over an ontology in a group meeting room” introduces some ways how to improve ontology engineering in a face-to-face situation. The last scenario “Creating an ontology with the help of software agents” is included to capture first thoughts on how software agents could help to create ontologies in an online-setting.

6.3.1 Learning the application

1. Justine wants to create a trial ontology project for learning how to use the application. Therefore she wants to create a small project on her desktop machine.
2. She opens up the application. She finds a screen which gives her different options such as “load an example ontology”, “create a new ontology development project”, “load an already existing ontology development project”, “import a new configuration file for a distributed ontology development project”, or “learn the application with a tutorial”. Justine wants to learn how to use the application. Therefore she starts the tutorial.
3. The tutorial guides her through the process of creating a new ontology development project.¹
4. A wizard dialog opens to guide Justine through the process of creating a new ontology development project. Justine is prompted to provide information where she wants to save her ontology development project. Justine chooses to save the ontology on her local hard drive as a file.
5. Then the tutorial explains briefly what an ontology is, what concepts, relations and instances are and how the application labels them. This tutorial part can be skipped. Justine skips it because she has read introductory literature on ontology engineering.

¹The steps for creating a new ontology without the tutorial are equivalent to the steps described here.

6. Now the tutorial explains in an interactive way how Justine can create new concepts, relations, and instances.
7. Then the tutorial shows Justine, how she can switch from the “ontology editing mode” of the application to the “argumentation mode”. Justine follows the tutorial and switches to the “argumentation mode”. In the argumentation mode Justine can only edit the ontology if she provides an argumentation. (see also scenario “Arguing over an ontology in an online-setting”)
8. In the next step the tutorial explains the argumentation model. The tutorial explains the differences between issues, ideas, and arguments and guides Justine through the process of creating issues, ideas and arguments. During the tutorial the application simulates the argumentation of another person.
9. The last step of the tutorial gives a brief overview about different representation formalisms and shows how to export the ontology in a specific ontology language like OWL.

The screen shots don’t show the “ontology editing mode”, but only the “argumentation mode”. The reason for this is that the ontology editing mode just provides the basic functionality of a standard ontology editor.

6.3.2 Setting up a new Ontology Development Project for Distributed Partners

1. Justine’s work group has decided to start a new ontology development project on European law. They have made already a feasibility study and have written an ontology requirements specification document (ORSO). They are now about to start the actual development of the ontology.
2. Justine is responsible for setting up the work environment, so that the project partners, which are located in different European cities, have access to the application. Justine contacts the system administrator Jack, who is responsible for the application and asks him to set up a new development projects together with a number of partners. Jack wants to know who the project owner will be and the e-mail address of each project partner. Justine’s direct boss is the project owner. Justine also promises to send an e-mail with the e-mail addresses of the other partners directly to Jack and they finish their phone call. Justine collects the e-mail addresses of the project partners and sends them to Jack. Actually all partners are able to add more persons to the development team, so that administration is less costly. Jack sets up the new project. He enters the e-mail addresses from Justine into a form. After everything is configured the application sends individual passwords and a configuration file to all participants. The configuration file includes machine-readable information for the clients on how to contact the ontology development server.

3. Justine received such a configuration file and a password. She saves the file and double clicks on it. The application client on her desktop machine opens and accesses the configuration file. The application client opens up a connection to the ontology engineering server. The application prompts Justine to enter her password and then she can start working.
4. The first step for Justine is to select the Microsoft Word document with the competency questions and import the competency questions into the application.
5. Now the real work for Justine and her colleagues starts. Their first task now is to cluster the competency question in different folders and to order the questions by their semantic distance within these groups². After this has been done the actual ontology development starts.

6.3.3 Arguing over an ontology in an online-setting

1. Justine and her research group have started the work on the ontology. The competency questions Justine imported are now sorted and the order is saved, so that it is preserved and available for every team member. They have created about 30 folders of competency questions with about 25 questions in each folder. The application is in the “argumentation-mode”. This means the ontology can only be modified, if an argument is given. The application allows in an online-setting only the “argumentation-mode”. Only the project owner can switch the development to the ontology-editing-mode.
2. Now they start to work on a set of competency questions in the folder on “domestic violence”. Justine triggers the command so that the application creates tasks for every competency question and the application assigns them equally to different persons in her research group. Justine is able to select those persons from a list of all project members, so that not everybody who is a project member will receive tasks. The selected team members are assigned to tasks automatically.
3. Now the team starts to raise issues and to create ideas on the competency questions they have been assigned to. Justine starts with her competency questions. After she finishes her work on one competency question she marks it for review. She can select either everybody on the research team as reviewers or only certain people. Additionally she can enter a brief message for the reviewers. The other persons in the group will then find a new task in their task-lists saying that they should review the issues and ideas Justine has created. The other persons in the team can see the changes Justine has made instantly and can directly respond to her raised issues and ideas.

²Based on the encouraging results described in chapter 2, we expect this step to become highly automated

4. When the others start to review the competency questions Justine had worked on, they can decide, if they agree with issues and ideas or not. If they agree with the issues and ideas, they click on the agree button, but they can also provide more “examples” or “evaluations & justifications”. If they disagree they can click on the disagree button and also start a discussion by giving “counter examples” and providing “alternatives & contrasts”. They can also add new issues and ideas as well as elaborations. To make their job easier, they can generate a visualization of the conceptualization Justin has suggested. If the visualization is modified, it is saved together with an “alternative & contrast” argument, a “counter example” argument, or an elaboration.
5. Justine makes a coffee break. She meets a colleague of hers at the coffee machine. They talk about a way how to model different concepts and relations in the domestic violence part of the ontology. After a while they agree on a couple of concepts. Justine and her colleague go to Justine’s office and enter a new issue and their ideas. They mark themselves both as creator of the issues and ideas, so that both of them will be notified (via tasks and either e-mail or IM).
6. On the next day Justine is automatically notified in an unobtrusive way (either via a daily mail or IM, and a new entry in her task-list in the application) that a couple of new arguments have been provided on her issues and ideas in the discussion. She looks at the different issues others have commented on. One issue has been discussed rather heavily. So she selects this issue. She now gets a visualization of the ontology that focuses on her first conceptualizations. Then she changes this initial visualization so that it shows the ways others proposed to visualize these concepts. She finds that she actually likes one of the other proposals better than her own. She votes for this other conceptualization. This will mark her idea as inactive because Justine as the owner of the idea has voted for a different way to conceptualize the raised issues. If another person now selects the issue she will not see Justine’s first conceptualization, but the conceptualization, which has the most positive vote rating (ratio of “agree”-votes to “disagree”-votes). The ideas for a concept are approved when either a defined amount of the team members have voted in favour for a concept or the project owner decides to use certain ideas.
7. Justine also reviews issues on other competency questions and provides arguments on issues and ideas of other people. She finds one issue that has been discussed very heavily. She makes a print out of this part of the ontology, which helps her to draw out different solutions. Finally she suggests to create two different elaborations to make the discussion more fertile. Justine additionally creates new tasks for the other team members to review her suggestions (The others receive a new task). After a while all the others have seen her arguments and agree. Justine’s boss as the ontology-owner therefore creates two new elaborations.
8. After some weeks they agree on how things should be conceptualized. They finally agree on the last few heavily discussed ideas. Now the ontology can be checked

by an ontology engineer to determine if it is formally correct. The ontology engineer points out a couple of issues, he needs help on from the domain experts. Then the first version of the ontology is created and can be exported to a representation formalism. Alternatively an ontology engineer could have taken part in the discussions. This would avoid invalid conceptualization of the domain and would allow skipping the last step. Also automatic evaluation tools would help in the formalization of the agreed on conceptualization.

6.3.4 Arguing over an ontology in a group meeting room with a beamer

1. Justine and two other people in her research group have been assigned to the task of creating a small knowledge base on international custom affairs in Spain. The development plan requires the creation of an ontology as an organization principle for a knowledge base. Today they have set up a meeting in a small meeting room. Justine has organized a beamer and connected it to her laptop. On her laptop she has already started a new ontology development project locally.
2. Justine also sets up a microphone in the middle of the table so that the application can record the discussion. The application also makes screenshots every few seconds to allow easy navigation through the recorded discussion afterwards because Justine can see in a movie file what she did on the screen and is able to listen again to the discussion that took place.
3. Justine and her two colleagues are now starting to create a new ontology. They have different sources from which they are taking ideas for the ontology engineering. These sources or links can be organized similarly to competency questions in folders. The sources can be short extracts copied & pasted into the application, various competency questions, or other documents.
4. They start out with discussing about a set of conceptualizations. Justine creates some new concepts and creates some relations among the concepts on her laptop. With the help of the beamer her two colleagues can see what she is doing. They all discuss about the best way to conceptualize the issue(s) at hand. Justine changes a couple of things once in a while. When they finally agree on a conceptualization Justine makes some notes on the pro and contra arguments within the discussion belonging to the set of conceptualizations. If they are discussing concepts that are too far away from their actual topic, Justin creates a new issue so that they can postpone the discussion. They are able to finish an initial version of the ontology in this group meeting.
5. After they have parted Justine refines the notes on the argumentation. She refers back to the recorded movie of the discussion once in a while to remember the argumentation again. She just records the arguments that are most important to her and

summarizes the groups' decisions. After she is satisfied with the documentation, she wants to set up an online discussion about the newly created ontology. This allows them to work asynchronously on the ontology and finish the development.

6. Justine asks the system administrator Jack, who is responsible for the application, to use the ontology development project she has locally on her machine as a starting point for a discussion in an online setting. Jack asks for the e-mail addresses of Justin's colleagues and the file of the ontology development project. Then Jack sets up a new ontology development project in an online-setting. Justin's colleagues get a configuration file via e-mail so they can load the project in their client of the application.
7. Now they can refine the ontology until they finally agree on all concepts.

The screen designs described later do not support the whole functionality described in this scenario. It is kept as an resource for future work.

6.3.5 Creating an ontology with the help of software agents

This scenario is used for scoping the application.

1. Justine and two other colleagues are starting an ontology development project. They are asked to create a topic hierarchy for the easy retrieval of all cases on trade law in the UK. As they expect the ontology to be rather large they plan to use a software agent with ontology learning capabilities. This software agent will take as its input all the cases on trade law that are available electronically.
2. Justine and her two colleagues actually start out by creating a couple of high-level concepts for trade law. They have a similar set up as in the scenario "Arguing over an ontology in a group meeting room with a beamer". The only difference today is that they only want to find some very basic concepts as input for the software agent. After the discussion has finished Justine documents the argumentation as described in the scenario "Arguing over an ontology in a group meeting room with a beamer".
3. Then Justine talks to the ontology engineer Richard. Richard has already prepared the documents with cases on trade law to be fed to the ontology learning algorithms. Justine passes Richard the ontology they have created as an initial version. Richard creates a new ontology development project in a distributed environment. Then Richard configures the software agent in cooperation with Justine. Richard chooses the different parameters the software agent should use. When everything is set up Richard starts the software agent.
4. The software agent starts to work on the text corpus. It creates suggestions for classes, relations, and instances, including some argumentation for why it created a conceptualization or a set of conceptualizations.

5. After the software agent has finished Justin and her colleagues separately view the different suggestions of the software agent. They start with the suggestions for concepts and eliminate suggestions that do not make sense at all. This also eliminates suggestions for relations between concepts that have been already eliminated. They also remove relations that do not make sense.
6. Then they start to discuss about the rest of the suggestions, again gathering in a meeting room. They go through the different suggestions of the software agent for concept, relations, and instances and use the arguments of the agent in favour or against the creation of new conceptualizations. The setting for this is similar to the setting as arguing over an ontology in a meeting room.
7. The draft ontology created by Justine and her colleagues then finally formalized by the ontology engineer Richard.

6.4 Platform Capabilities and Constraints

The client will use the eclipse platform. The only constraints are those, which apply when designing for a normal workstation. A server platform is not within the scope of this document except that it needs to support a near real-time propagation of “ideas”.

6.5 Usability Goals

The project focuses on ease of learning rather than on ease of use. This means that a user should learn the application easily. In decisions where trade offs between ease of learning or ease of use had to be made in the design, ease of learning was preferred.

As a measurable goal the application should be more efficient to use than an off the shelf wiki for discussing the modelling of an ontology.

6.6 Work Model

The work model of the application follows the roughly laid out process in figure 6.1. At first the requirements for the ontology are gathered and sources from which ontological knowledge can be extracted are collected. These can be for example competency questions or all kinds of other documents that can be referred to by an URI. These so called sources are imported into the application allowing to work with them in an integrated way, as well as the tracking of the design decisions in later stages of the ontology engineering process. When the sources are imported into the application tasks are created and distributed to the members of the work team to look at each source and create issues

regarding the sources. This ensures that all sources are taken into account. Work can be done in parallel, meaning that several people can create sources, tasks, issues, ideas, and arguments simultaneously. When the issues are created the team will then create ideas how to solve issues. An idea is always a possible ontology change operation. An idea is created by provisionally adding a modelling primitive. If ideas are contradicting then a new idea set can be created removing some modelling primitives and replacing them by other provisional modelling primitives. On issues and ideas the team can exchange arguments on how to actually design the new ontology. The actual decision if modelling primitives should be part of the new ontology is determined either by “democratic” voting of the team members or by the ontology owner, who then approves one idea or idea set. In case that an already approved idea creates a conflict a new issue needs to be raised, ideas need to be discussed and the conflict needs to be resolved. All these process steps are managed by tasks. The outcome of the process is a new shared ontology.

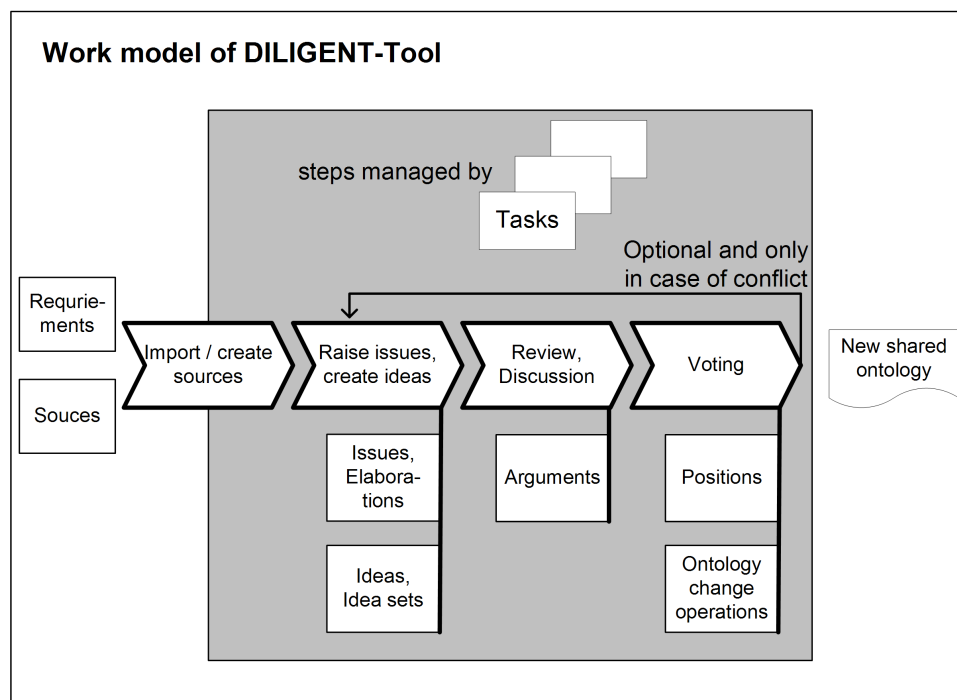


Figure 6.1: Work model of the DILIGENT-Tool

Appendix A describes in detail the windows of the EvA tool. Appendix B offers a glossary of the most important terms used within the EvA context.

6.7 Conclusions

In the following you may find our results of the evaluation of the presented design, a list of open issues, and a first glimpse of the ongoing implementation.

6.7.1 Evaluation

The application design has been evaluated in an informal setting. Two scenarios were created from the screen shots above to test the design of the prototype. The stakeholders were taken through a prototype fulfilling the following two tasks:

Scenario 1:

- Open first task
- Create a new issue
- Create the concept Partnership
- Create the concept Marriage as subconcept of Partnership
- Create the concept Human
- Create the concept Man as subconcept of Human
- Create the concept Woman as subconcept of Human
- Create the relation isMarriedIn. Domain: Man, Woman, Range: Marriage
- Create the concept RegisteredPartnership as subconcept of Partnership
- Create the relation isPartnerIn. Domain: Human, Range: RegisteredPartnership

Scenario 2:

- Open first task
- Provide a challenge for the issue.
- Provide a challenge for the idea RegisteredPartnership

The results of the evaluation have been fed back already into the prototype design. During the evaluation issues like users can only vote once, meaning that they cannot change their mind, were uncovered and resolved in the redesign of the prototype.

6.7.2 Open issues

It remains an open issue what kind of voting mechanism to use. The DILIGENT methodology suggests that a board decides whether an idea shall be implemented or not. The tool design offers a faster, but maybe less reliable way based on voting by all team members. Probably different voting mechanisms have to be applied depending on the ontology at hand, so the final tool should be agnostic towards this issue.

It is also interesting to see if an automatic classification of arguments can be applied: is an argument for or against an idea? Is an example provided? The results in Human Language Technologies and Knowledge Discovery seems promising towards such a goal.

Another issue is how to integrate software agents in the ontology engineering by employing ontology learning algorithms. Although this is clear from the DILIGENT methodology [STV⁺05], it is not yet settled how to implement this within the tool.

6.7.3 Implementation

After reviewing the tool design by case study partners, now an implementation of EvA is ongoing. It is based on Eclipse and OntoStudio, and will follow closely the design outlined in this chapter. We expect the tool to be evaluated within SEKT in the next year for the further evolution of the SEKT ontologies.

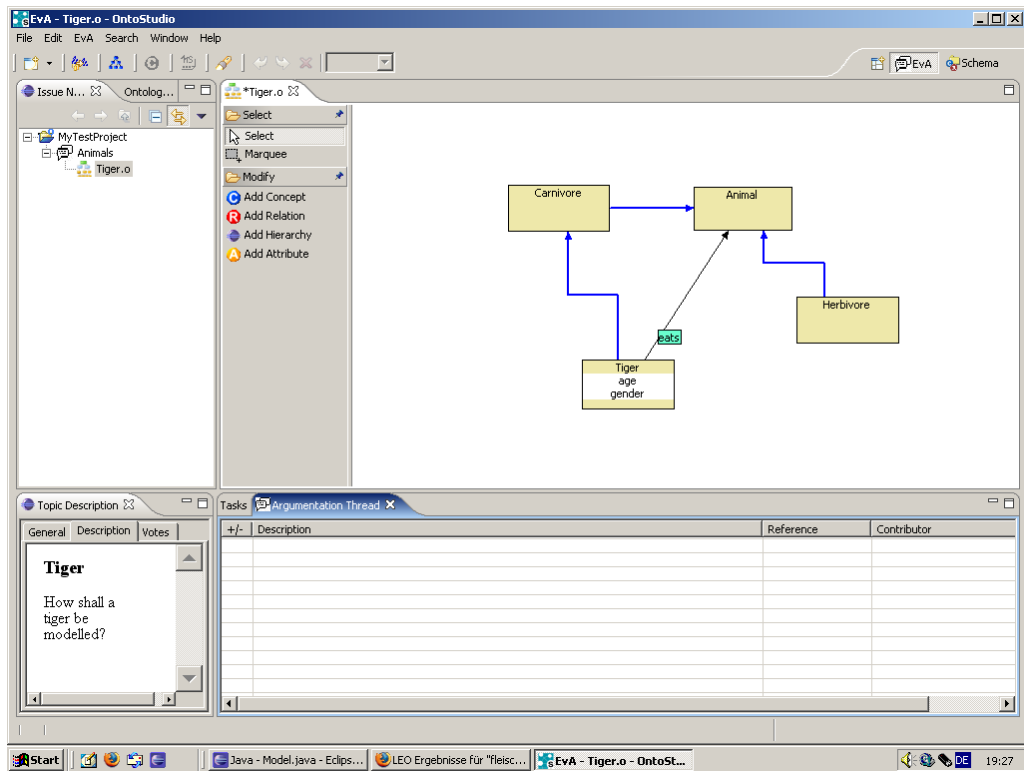


Figure 6.2: Screenshot of EvA

Chapter 7

Wiki-based ontology population

Whereas the tool described in the previous chapter is geared towards the creation of the classes and properties of an ontology – or the T-Box, to speak in Description Logic terms – the wiki-based tool described in this chapter aims at providing an easy mean for the population of an ontology. Staying true to the DILIGENT process we aim at a distributed setting and for the easy capturing of arguments.

Most ontology engineering environments today provide means to also create instances to populate the ontology. But the user interface is not easy to use for non-technical users. So even after the ontology has been designed, populating the ontology – this means, filling the ontology with instances and data – usually needs a full fledged ontology engineering environment installed at the end users desktop.

Already in the first year of SEKT we identified wikis as an easy to use technology for collaborative working [TSVP04, CL01]. But we had to learn that using it for evolving and engineering ontologies simple off-the-shelf wikis are not appropriate, as shown in chapter 5. Still, the users never complained about the wiki itself as being too complicated to use – they always pointed out certain deficiencies of the wiki, like not being alerted or not being able to interoperate with other tools properly.

So again we chose a wiki as a base for a DILIGENT inspired tool. In this chapter we describe the background of the chosen wiki, the design of the extensions, possible usage scenarios and the state of the implementation.

7.1 MediaWiki and Wikipedia

The MediaWiki software is the wiki software behind the hugely popular Wikipedia¹ website. MediaWiki² was developed by the Wikipedia community especially for the Wikipe-

¹<http://www.wikipedia.org>

²<http://www.mediawiki.org>

dia, but is now used in several other websites as well.

Wikipedia is a collaboratively edited encyclopaedia, available under a free licence on the web. It was created by Jimbo Wales and Larry Sanger in January 2001, and has attracted some ten thousand editors from all over the world. As of 2005, Wikipedia consists of more than 2,5 million articles in over two hundred languages, with the English, German, French and Japanese editions being the biggest ones [Wal05]. The idea of Wikipedia is to allow everyone to edit and extend the encyclopaedic content (or simply correct typos).

The used software thus obviously fulfils the requirements towards easy usability, scalability, enabling geographically distributed collaboration and availability. It is also actively maintained. Wikipedia has furthermore developed a sophisticated conflict resolution process, based on discussion pages on every article, talk pages and social processes.

In order to leverage these capabilities we must carefully design and implement our extensions to the MediaWiki software. The extension was first proposed to the Wikipedia community and is discussed since then [KVV05].

7.2 Semantic extensions to MediaWiki

We expose Wikipedia's fine-grained human edited information in a standardised and machine-readable way by using the W3C standards on RDF [MM04], XSD [FW04], RDFS [BG04], and OWL [SWM04]. This not only opens new ways for improving Wikipedia's capabilities for querying, aggregating, or exporting knowledge, based on well-established Semantic Web technologies, but also helps in enabling users to easily create further knowledge bases.

Besides the usage within the SEKT project, we aim at adding the semantic extension to Wikipedia. Then we hope that Semantic Wikipedia can help to demonstrate the promised value of semantic technologies to the general public, e. g. serving as a base for powerful question answering interfaces.

Our primary goal is to provide an extension to MediaWiki which allows to render important parts of the knowledge within the wiki machine-processable with as little effort as possible. This specific goal also creates a number of challenges that one has to be aware of.

First and foremost, extensions of wikis have to satisfy highest requirements on usability, since the large community of volunteers is the primary strength of any wiki. Users must be able to use the extended features without any technical background knowledge or prior training. Furthermore, it should be possible to simply ignore the additional possibilities without major restrictions on the normal usage and editing. Another important aspect of using a wiki is that the community must always have full control, and that users can freely modify and extend the content. We start with an overview of the kind of se-

mantic information that is supported, and proceed by discussing *typed links* and *attributes* individually.

7.2.1 The Big Picture

To ensure seamless integration with the common usage experience in a wiki, we approach our goal by slightly extending the way of creating a hyperlink between articles. As for the Web in general, links are arguably the most basic and also most relevant markup within a wiki, and their syntactical representation is ubiquitous in the source of any Wikipedia article. Through a minor, optional syntax extension, we allow wiki users to create *typed links*, which express a *relation* between two pages (or rather between their respective subjects). Below, we describe how this additional information can be added in an unobtrusive and user-friendly way.

Just like a normal link, a typed link is a link from one wiki page to another wiki page. A typed link simply additionally has a *type* which states the kind of relation between concepts. As an example, one could type a link from the wiki page “London” to “England” with the relation “capital of”. Even very simple search algorithms would then suffice to provide a precise answer to the question “What is the *capital of England*?” In contrast, the current text-driven search returns only a list of articles for the user to read through.

In order for such extensions to be used by editors, there must be new features that provide some form of *instant gratification*. Semantically enhanced search functions improve the possibilities of finding information within the wiki and would be of high utility to the community. But one also has to assure that changes made by editors are immediately reflected when conducting such searches. Additionally, the wiki’s machine-readable knowledge is made available for external use by providing RDF dumps. This allows for the creation of additional tools to leverage the wiki’s contents and re-use it in other contexts. Thus, not only humans may read the articles, but also a new range of services is enabled inside and outside the wiki due to the machine readable export.

To further improve the utility of our approach, we provide means to make other information explicit as well. First of all, MediaWiki’s category system suggests itself for machine processing as well. At the moment, categories are used solely for browsing. Advanced searching, possibly incorporating the given hierarchy, or even simple Boolean operations are not supported for categories. We map the categories to concepts in a given ontology, thus turning the wiki pages themselves to instances.

For another interesting source of machine readable data, we also incorporate the data values a wiki can hold. Typically, such values are provided in the form of numbers, dates, coordinates, and the like. For example, one would like to obtain access to the population number of London. It should be clear that it is not desirable to solve this problem by creating a semantic link to an article entitled “7421328” because this would create an unbearable amount of mostly useless number-pages whereas the textual title does not even capture the intended numeric meaning faithfully (e.g. the natural lexicographic

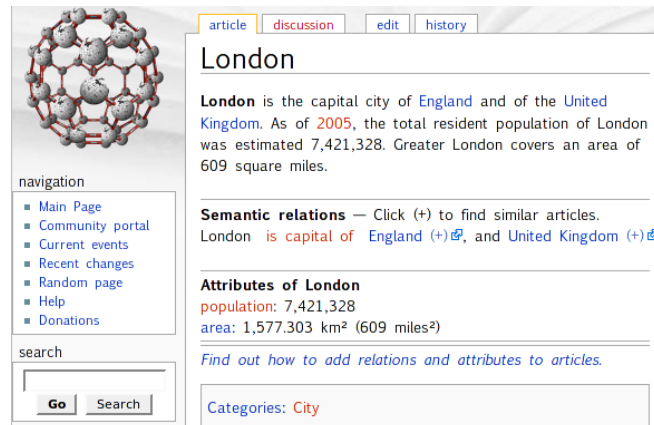


Figure 7.1: A semantic view of London.

order of titles does not agree with the natural order of numbers). Therefore, we introduce an alternative markup for describing attribute values in various datatypes. Moreover, we address the problem of handling units of measurement that are often given for data values.

To summarise these features, consider the example depicted in Figure 7.1. Here you see a short article on London. The markup contained in the article source allows to extract key facts from this text, which are displayed below the article. In the following sections, we explain in detail how this is achieved by an editor.

7.2.2 Usage of Typed Links

A regular hyperlink already states that there is some relation between two pages—an information that search engines like Google successfully use to rank pages in a keyword search scenario. Typed links can even go beyond that, since they are interpreted as *semantic relations* between two concepts described within articles. Here, instead of discussing the “URI crisis” again, we can assume that a wiki page URI unambiguously represents the concept discussed on that page and not the page itself. We remark that not every link should be turned into a typed link: many links do not have an explicit meaning and serve merely navigational purposes. The utility of typed links stems from their ability to distinguish links that are “meaningful” in a given context.

The suggested typing mechanism can be understood as a kind of categorisation of links. Like in the case of categories and articles, wiki authors are free to employ arbitrary descriptive labels for typing links. To do so, one has to extend the syntactical description of a hyperlink. Without semantic extensions, the source of the article in Figure 7.1 looks as in Figure 7.2. In MediaWiki, links to other articles are created by simply enclosing the article names in double brackets. Classification in categories is achieved in a similar fashion by providing a link to the category. However, category links are always displayed in a special format at the bottom of the wiki page, without regard to the place of the link

```
'''London''' is the capital
city of [[England]] and of the
[[United Kingdom]]. As of
[[2005]], the total resident
population of London was
estimated 7,421,328. Greater
London covers an area of 609
square miles.
[[Category:City]]
```

Figure 7.2: Source of an article on London using Wikipedia’s current markup.

inside the text.

Now in order to explicitly state that London is the capital of England, one just extends the link to `[[England]]` by writing `[[capital of::England]]`. This states that a relation “capital of” holds between “London” and “England.” Typed links therefore stay true to the wiki-nature: every user can add a type to a link or change it. It should be clear that the textual labels of a link type can be chosen arbitrarily by the user. Of course, in order to make improved searching and similar features most efficient, the community will settle down to re-use existing link types. As in the case of categories, we allow for the creation of descriptive pages on link types to aid this process.

In the rare cases where links should have multiple types, users can write `[[type1::type2::...::typen::target article]]`. Sometimes it is desirable that the displayed text of a hyperlink is different from the title of the article it links to. In MediaWiki, this can be achieved by writing `[[target article|link text in article]]`, and this option is not affected by any of the syntactical extensions we allow for specifying the target article.

Note how typed links integrate seamlessly into current wiki usage. In contrast to all other semantic wikis we are aware of, Semantic MediaWiki places semantic markup directly within the text and thus ensures that machine-readable data agrees with the human-readable data within the article. The order of writing relation and linked article in our notation makes the extended syntax largely self-explicatory (provided that labels are chosen carefully).

7.2.3 Attributes and Types

Data values often play a crucial role within a knowledge base, and machine access to this data yields numerous additional applications. In contrast to the case of links, attribute values are usually given as plain text, so that there is no such straightforward syntactical extension for marking this information. Yet we settled down to introduce a syntax that is very similar to the above markup for typed links. Namely, in order to make, e.g., the value

for the population of London explicit, one writes `[[population:=7,421,328]]`. Using `:=` instead of `::` allows for an easy distinction between attributes and typed links, while also reflecting the propinquity of these concepts.

In spite of these advantages, we are aware that introducing link syntax for parts of text that do not create hyperlinks might be a possible source of confusion for users. But the fact that most characters are allowed inside MediaWiki articles severely restricts the syntactical options. Therefore, MediaWiki also uses link syntax for denoting category membership, for relating articles across languages, and for including images, each of which does not create a normal hyperlink at the place of the markup. We thus believe that our choice is tenable, but we also allow to syntactically encapsulate annotations into MediaWiki's *template* mechanism, as described in Section 7.2.4.

Besides this, attributes behave largely similar to typed links from a user perspective. In particular, we enable users to freely choose names for new attributes and to provide alternative texts with the markup. The latter feature is often helpful, e.g. in order to write “London has a population of `[[population:=7,421,328|around 7.5 million]]`.”

Combining this information with the semantic data discussed above, the system should be able build a list of all European cities ordered by their population on the fly. To enable this, we also need a powerful query language and tools that support it. Fortunately, such languages have been developed for various explicit representations of knowledge, SPARQL³ being the most recent outcome of these developments. Disclosing these achievements to our system requires us to describe semantic data in RDF and to provide an appropriate storage architecture. Another challenge is to develop user interfaces that grant access to such powerful query mechanisms without requiring knowledge of SPARQL syntax. Luckily, we can address this problem gradually, starting with intuitive special-purpose interfaces that restrict expressivity. Section 7.4 sketches how our system supports the creation of such tools in a way that is mostly independent from our implementation.

So far, attributes appear to be quite similar to typed links, but there are a number of further issues to be taken into account. For example, a statement like “Greater London has a total area of 609” does not mean anything. What is missing here is the *unit of measurement*. To solve this problem, our current implementation provides generic support for such units in a fully intuitive way. In the present example, the user would just write “`[[total area:=609 square miles]]`,” though the unit could also be denoted as “mi²,” “sq mile,” and much more. Observe that there are often multiple identifiers to denote the same unit, but that there are also different units to measure the same quantity. Therefore, the system offers support for multiple unit identifiers. In a growing number of cases, the system provides automatic conversion of a value to various other units as well, such that searches can be conducted over many values irrespective of their unit. This allows users to state their values either in square miles or square kilometers.

³<http://www.w3.org/TR/rdf-sparql-query/>

```
'''London''' is the capital
city of [[capital of::England]]
and of the [[capital of::United
Kingdom]]. As of [[2005]], the
total resident population of
London was estimated
[[population:=7,421,328]].
Greater London covers an area
of [[total area:=609 square
miles]]. [[Category:City]]
```

Figure 7.3: Source of an article on London with semantic extensions.

For the user, these features are completely transparent: she just enters the unit of her choice. Of course, the degree of support varies among units. Users also receive the immediate benefit of having converted values displayed within the article. Thus the markup given in Figure 7.3 does indeed produce the output of Figure 7.1. Finally, if a unit is unknown, the (numerical) value is just processed together with its (textual) unit identifier. Note that the rendered value within the text of a wiki page is never affected by any automatism and always displayed as entered.

Due to the fact that units can have ambiguous identifiers (e.g. “ml” for “miles” and for “millilitres”), users must be able to state which kind of units are supported for an attribute. Many features, such as sorting results according to some attribute, also require knowledge about its basic datatype (integer, float, string, calendar date, ...). Details on how unit support is implemented, and on how users can supply the required type information are given in Section 7.3 below. Here, we only remark that users will usually find a sufficient amount of existing attributes, so that they do not have to bother about their declaration. Furthermore, many different attributes behaves completely similar with respect to the type of data and the supported units. For example, the declaration of `total area` can be reused for `water area`, `surface area`, `acreage`, and many other kinds of areas.

7.2.4 Semantic Templates

With the above features, the system is also able to implement a technology sometimes referred to as *semantic templates*. MediaWiki already offers a template mechanism that allows users to include predefined parts of text into articles. Templates can also include placeholders which are instantiated with user-supplied text when the template is included into an article. This feature allows to put varying content into a fixed format, and was mainly introduced to ensure a consistent layout among articles. However, by simply adding typed links or attributes to the template text, our implementation allows to employ templates for encapsulating semantic annotation. In some cases, one could even modify existing templates to obtain a large amount of semantic data without changing any article.

Yet, many existing uses of templates disallow such semi-automatic annotation. Indeed, placeholders in templates are often instantiated with short descriptive texts, possibly containing multiple links to other articles, such that no single entity can be annotated. Yet, semantic templates are a valuable addition to our approach, since they can simplify annotation in many cases.

7.3 Design

In this section, we devise an architecture for a concrete implementation. We start by introducing the overall format and architecture for data storage, and continue with discussing the workflow for evaluating typed links and attributes given in articles. For attributes, this includes declaration, error handling, and unit conversion.

Seeking a suitable formal data model, we notice the close resemblance of our given input data to RDF and RDFS. Typed links basically describe RDF properties between RDF resources that denote articles, attributes correspond to properties between articles and RDF data literals, and MediaWiki's current classification scheme can be modelled with RDF-classes. Note that MediaWiki already restricts the use of categories to the classification of articles—there are no categories of categories. In other words, we are dealing with a fragment of RDFS that is largely compatible with the semantics of OWL DL, and we might choose representation in either formalism without major semantic obstacles.

These considerations enable us to view our system as a convenient tool to author RDF specifications, removing technical barriers to allow cooperative editing by a large number of users.

The availability of mature software for processing and storing RDF is indeed very important. First of all, it helps us to retain the scalability provided by the current MediaWiki software and showcased by Wikipedia itself. Specialised RDF databases (“triplestores”) available today should be able to deal with the expected loads⁴, and do even provide basic reasoning support for RDFS constructs. Furthermore, recent RDF-tools provide simplified data access via query languages like SPARQL which can be used to implement search interfaces. RDF software libraries facilitate the task for external developers who want to reuse semantic data exported from MediaWiki in an XML-based RDF serialisation. The overall architecture we envision is pictured in Figure 7.4, which we will consider in more detail in Section 7.4. Within the SEKT project the fastest triple store to date was developed, OWLIM⁵ by OntoText, and we hope to be able to integrate it into our prototype.

⁴English Wikipedia *currently* counts 800,000 articles, but it is hard to estimate how many RDF triples one has to expect. High numbers of parallel requests yield another invariant often ignored in benchmarks of current stores.

⁵<http://www.ontotext.com/owlim/>

7.3.1 Typed Links

In the following, we explicate how the above ideas can be substantiated in a concrete architecture that allows input, processing, and export of typed links in MediaWiki. This encompasses the treatment of categories, attributes, and schema information as well, though some additional measures are required for some of these cases.

Section 7.2 explained how users can include type information into existing links. According to our above discussion on the relationship to RDF, the type that a user gives to a link is just a string that serves as a label for an RDF property. Thus, proper encoding in RDF can be achieved without requiring users to declare types of links in advance, and arbitrary names can be introduced by users.

In spite of these additions, the processing of typed links entered by a user does not involve information about relations. Hence only local data sources need to be accessed for transforming typed links into RDF-triples that are sent to the triplestore which serves as the storage backend. To assure that the triplestore contains no information that is not contained in the text of the article, we must delete outdated triples prior to saving new information. By design, every wiki page affects exactly those triples that involve this page as the subject, so that we just have to delete these triples on update. Note how crucial this *locality* of our approach is for obtaining a scalable system. Summing up, any user-triggered update of a page leads to the following three steps:

1. Extraction of link types from the wiki text (parsing).
2. Transformation of this semantic information into RDF triples.
3. Updating the triplestore by deleting old triples and storing new ones.

Each of these operations can be performed with a minimum of additional resources (computation time, bandwidth), and the update of the triple store could even be achieved asynchronously after the wiki text was saved. Edit conflicts by concurring changes are detected and resolved by the MediaWiki software.

Our architecture also ensures that the current semantic information is always accessible by querying the triple store. This is very convenient, since applications that work on the semantic data, including any search and export functions, can be implemented with reference to the storage interface independently from the rest of the system. In Section 7.4, we give some more details on how this is achieved. Here we just note the single exception to this general scheme of accessing semantic data: the infobox at the bottom of each article is generated during parsing and does not require reading access to the triplestore. This is necessary since infoboxes must be generated even when previewing articles *before* they are actually saved, but of course it also reduces overhead.

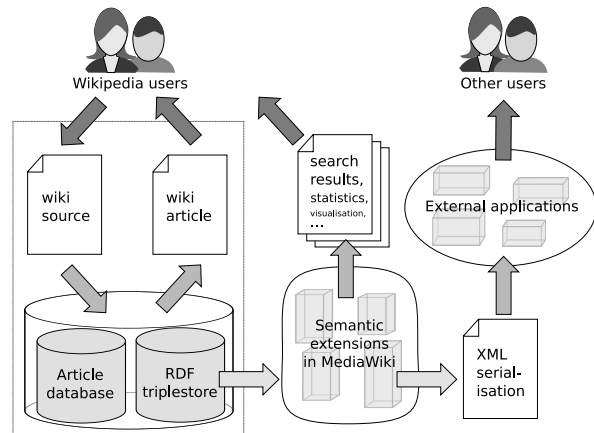


Figure 7.4: Basic architecture of the semantic extensions to MediaWiki. On the left, Wikipedia users edit articles to enter semantic information (see Section 7.3). Further extensions to MediaWiki use this data and export it to external applications (see Section 7.4).

7.3.2 Attributes

Attributes are similar to typed links, but describe relationships between the subject of the wiki page and a (possibly complex) data value, instead of between two pages. With respect to the general architecture of storage and retrieval, they thus behave completely similar to typed links. But additional challenges for processing attribute values have already been encountered in Section 7.2. In this section, we formally relate the lexical representation of attribute data to the data representation of XML Schema. This leads us to consider the declaration of datatypes for semantic attributes, before focussing on error handling and the complex problem of unit support.

Value Processing and Datatypes

A primary difficulty with attributes is the proper recognition of the given value. Since attribute values currently are just given as plain text, they do not adhere to any fixed format. On the other hand, proper formatting can be much more complicated for a data value than for a typed link. A link is valid as soon as its target exists, but a data value (e.g. a number) must be *parsed* to find out its meaning. Formally, we distinguish the *value space* (e.g. the set of integer numbers) from the *lexical space* (e.g. the set of all acceptable string representations of integers). A similar conceptualisation is encountered for the representation of datatypes in RDF, which is based on XML Schema (XSD) [FW04].

We arrive at a system that consists of three major components: *data values* (e.g. “3,391,407”) are assigned to *attributes* (e.g. “population”) which are associated with a given *datatype* (e.g. “integer”). To allow users to declare the datatype of an attribute, we

introduce a new namespace “Attribute:” that contains pages on attributes. Within these pages, one can provide human-readable descriptions as in the case of relations and categories, but one can also add semantic information that specifies the datatype. To minimise the amount of new syntactic constructs, we propose to use the concept of typed links as introduced in this paper. Therefore we reserve a third (and last) namespace for types: “Type:”. Then, using a relation with built-in semantic we can simply write

```
[ [hasType::Type:integer] ]
```

to denote that an attribute has this type. The declaration of datatypes can also be facilitated by templates as introduced in Section 7.2.4. For instance, one can easily create a template that provides the semantic information that an attribute is of type integer, but that also includes human-readable hints and descriptions.

The types themselves are built-in, and their articles only serve to give a description of proper usage and admissible input values. If need arises, one could also conceive a system to customise types by giving additional descriptions inside the pages in the “Type:”-namespace.

Note that the general workflow of processing attribute input becomes slightly more complex than in the case of typed links. Before extracting the attribute values from the article source, we need to find out about the datatype of an attribute. This requires an additional reading request to the storage backend and thus has some impact on performance. Since other features, such as template inclusion, have similar requirements, we are optimistic that this impact is tenable.

Error Handling

In contrast to typed links, the handling of attributes involves cases where the system is not able to process an input properly, although the article is syntactically correct. Such a case occurs whenever users refer to attributes for which no datatype was declared, but also if the provided input values do not belong to the lexical space of the datatype.

A usual way to deal with this is to output an error that specifies the problem to the user. However, in a wiki-environment, it is vital to tolerate as many errors as possible. Reasons are that technical error messages might repel contributors, that users may lack the knowledge for understanding and handling error messages, and, last not least, that tolerating (temporal) errors and inconsistencies is crucial for the success of collaborative editing. Thus our system aims at catching errors and merely issuing warning messages wherever possible.

For the case of missing datatype declarations, this can be achieved by presuming some feasible datatype based on the structure of the input. Basically, when encountering a numerical value, the input is treated as floating point number. Otherwise it is processed as a string. A warning inside the infobox at the article bottom informs the user about the possible problem. Here we exploit that RDF includes a datatype declaration for each

value of a property. A property can thus have values of multiple datatypes within one knowledge base. This is also required if an existing datatype declaration is changed later on (recall that anyone can edit declarations). The new type only affects future edits of articles while existing data is still valid.

If the datatype is specified but the input does not belong to the supported lexical space, we do not store any semantic information and restrict to issuing a warning message within the infobox.

Units of Measurement

The above framework provides a feasible architecture for treating plain values, like the number of inhabitants of some city. However, many realistic quantities are characterised not only by a numerical value, but also by an associated unit. This is a major concern in an open community where users may have individual preferences regarding physical units. Units might be given in different systems (kilometres vs. miles) or on different scales (nanometres vs. kilometres). In any case, reducing such input to a mere number severely restricts comparability and thwarts the intended universal exchange of data between communities and languages. Using different attributes for different units is formally correct, but part of the problem remains (values of “length (miles)” and “length (kilometres)” remain incomparable to RDF tools).

Our solution to this problem is twofold. On the one hand, we provide automatic unit conversion to ensure that large parts of the data are stored in the same unit. On the other hand, we recognise even those units for which no conversion support is implemented yet, so that we can export them in a way that prevents confusion between values of different units. To this end, note that it is fairly easy to separate a (postfixed) unit string from a numerical value. Given both the numerical value and the unit string, one can unambiguously store the information by including unit information into attribute (property) names. For example, the input `[[length:=3km]]` is exported as value 3 of a property identified as “length#km”. Users can freely use new units, and exported data remains machine-processable.

Since the power of semantic annotation stems from comparing attributes across the database, it is desirable to employ only a small number of different units. Users can achieve this manually by converting values to some standard unit and giving other units as optional alternative texts only. We automate this process by providing built-in unit conversion for common units. From the RDF output it is not possible to tell whether a unit conversion has taken place automatically, or whether the user has provided the value in the given unit right away. This has the further advantage that one can safely add unit support gradually, without affecting applications that already work with the exported data. Built-in unit support also allows us to provide automatic conversions inside the wiki page or infoboxes, which provides immediate gratification for using attributes.

7.4 Implementation

Important parts of the architecture from Section 7.3 have already been implemented. Readers who want to touch the running system are pointed to our online demo at wiki.ontoworld.org and to the freely available source code.⁶ We would like to encourage researchers and developers to make use of the fascinating amount of real world data that can be gathered through Semantic MediaWiki and to combine it with their own tools.

At the moment, Semantic MediaWiki is still under heavy development, and many features are just about to be implemented. Like MediaWiki itself, the system is written in PHP and uses a MySQL database. Instead of directly modifying the source code of the wiki, we make use of MediaWiki's *extension* mechanism that allows developers to register functions for certain internal events. For this reason, the Semantic MediaWiki extension is largely independent from the rest of the code and can be easily introduced into a running system.

Moreover, interested readers can easily implement their own extensions to the semantic extension. The general architecture for adding extensions is depicted in Figure 7.4. The box on the left represents the core functions of editing and reading, implemented according to the description in Section 7.3. As shown in the Centre of Figure 7.4, the obtained (semantic) information can be exploited by other extensions of MediaWiki by simply accessing the triplestore. Functions for conveniently doing so are provided with our implementation. Our current effort comprises some such extensions, specifically a basic semantic search interface and a module for exporting RDF in an XML serialisation. Additional extensions, such as improved search engines, can be added easily and in a way that is largely independent from our source code. MediaWiki provides means of adding "Special:" pages to the running system in a modular way, so that a broad range of semantic tools and interfaces could be registered and evaluated without problems.

Finally, the data export (as well as possible dumps of the database) can be utilised by independent external applications. Programmers thus obtain convenient access to the results of any MediaWiki-based cooperative online project. The possibilities this technology offers to enhance desktop applications and online services are immense – we discuss some immediate application scenarios in the section 7.5.

7.4.1 User Experience

We already saw that end users experience only small changes in form of some easy to grasp syntax, appearing at various places within articles. Here, we discuss some other changes that users will encounter in a Semantic MediaWiki.

Most prominently, users now find an infobox for semantic information at the bottom

⁶See <http://sourceforge.net/projects/semmediawiki>.

of each page. It helps editors to understand the markup, since it immediately shows how the system interprets the input. But the infobox also provides extended features for normal users. As shown in Figure 7.1, typed links are augmented with quicklinks to “inverse searches.” For example, if the article about *Hyde Park* states that it is located in London, a single click in the infobox suffices to trigger a search for further things located in London.

Additionally, data values can be connected with special features based on their datatype. For instance, articles that specify geographic coordinates can be furnished with links to external map services⁷. In the case of calendar dates, links could refer to specialised searches that visualise a timeline of other events around the given date.

Of course it is also possible to conduct searches directly. New special pages for “*Semantic Search*” will allow users to pose advanced queries. Providing user interfaces for this task is not at the core of our current work, but the system already includes a simple search for articles that have a certain relation to a given (set of) other articles. We provide means for developers to add their own search interfaces in an easy way, and we expect that many customised searches to appear (both experimental and stable, both within a Semantic MediaWiki system and on external sites).

7.5 Usage scenarios

We will pursue to apply the Semantic MediaWiki extension as presented here to Wikipedia, thus promoting SEKT technology outside the boundaries of the SEKT project. In this section we will discuss the possible usage scenarios of a Semantic Wikipedia.

Wikipedia is the biggest collaboratively created source of encyclopaedic knowledge. Growing beyond the borders of any traditional encyclopaedia, it is facing new problems of knowledge management: The current excessive usage of article lists and categories witnesses the fact that 19th century content organization technologies like inter-article references and indices are no longer sufficient for today’s needs.

Enabling even casual users to participate in the creation of an open semantic knowledge base, Wikipedia has the chance to become a hitherto unknown resource of semantic statements regarding size, scope, openness, and internationalisation. These semantic enhancements bring to Wikipedia the benefits of today’s semantic technologies like more complex or more specific ways of searching and browsing. Also, the RDF export, that gives direct access to the formalised knowledge, opens Wikipedia up to a wide range of external applications, that will be able to use it as a background knowledge base.

A wide range of applications becomes possible on the basis of a semantically enhanced Wikipedia. Here, we briefly outline the diversity of different usage areas, ranging

⁷Experimental versions of such services are already provided in English Wikipedia, but the focus there is not on further usage of the data.

from the integration of Wikipedia’s knowledge into desktop applications, over enhanced *folksonomies*, to the creation of multilingual dictionaries. Moreover, automated access to Wikipedia’s knowledge yields new research opportunities, such as the investigation of consensus finding processes.

Many desktop applications could be enhanced by providing users with relevant information from Wikipedia, and it should not come as a surprise that corresponding efforts are already underway.⁸ For instance, the *amaroK* media player⁹ seamlessly integrates Wikipedia articles on artists, albums, and titles into its user interface, whenever these are available. With additional semantic knowledge from Wikipedia, many further services could be provided, e.g. by retrieving the complete discography of some artist, or by searching the personal collection for Gold and Platinum albums. Similarly, media management systems could answer domain-specific queries, e.g. for “music influenced by the Beatles” or “movies that got an Academy Award and have a James Bond actor in a main role.” But the latter kind of question answering is not restricted to media players: educational applications can gather factual data on any subject, desktop calendars can provide information on the current date, scientific programs can visualise Wikipedia content (genealogical trees, historical timelines, topic maps, ...), imaging tools can search for pictures on certain topics—just to name a view.

These usage scenarios are not restricted to the desktop. A web-based interface does also make sense for any of the above services. Since Wikipedia data can be accessed freely and without major legal restrictions, it can be included in many web pages. Cooperations with search engines immediately come to mind, but also special-purpose services can be envisaged, that may augment their own content with Wikipedia data. A movie reviewer could, instead of adding the whole data about the movie herself, just use a template and integrates it on her website, pulling the data live from Wikipedia. Finally, portals that aggregate data from various data sources (newsfeeds, blogs, online services) clearly benefit from the availability of encyclopaedic information too.

On the other hand, Wikipedia can also contribute to the exchange of other information on the web. Folksonomies [Mat04] are collaborative tagging or classification systems that freely use on-the-fly labels to tag certain web resources like websites (del.icio.us¹⁰), pictures (flickr¹¹), or even blog entries [GH06]. In these applications, tagging simply means assigning a keyword to a resource, in order to allow browsing by those labels. These keywords distinguish neither language nor context. For example, the tag “reading” could either mean the city, *Reading, MA*, or the act of reading a book; the tag “gift” could either mean the German word for poison or the English synonym for present. On the other hand, different tags may mean mostly the same thing, as in the case of “country,” “state,” and “nation.” Searching for either one of them does not yield results tagged with the other, although they would be as relevant. A semantic tagging system, however, could

⁸http://meta.wikimedia.org/wiki/KDE_and_Wikipedia.

⁹<http://www.amarok.org>

¹⁰<http://del.icio.us>

¹¹<http://www.flickr.com>

offer labels based on meaning rather than on potentially ambiguous text strings. This task is simplified when using concept names and synonyms retrieved from a semantically enhanced Wikipedia in the user's chosen language. Similarly, Wikipedia's URIs can serve as a universal namespace for concepts in other knowledge bases and ontologies.

Semantic Wikipedia could also act as an incubator for creating domain ontologies, by suggesting domains and ranges for roles, or applicable roles for a certain concept (if the majority of countries seem to have a head of state, a system could suggest to add such a role to the ontology). Semantic Wikipedia also could be queried in order to populate ontologies quickly (one could, for example, pull all soccer players for their soccer ontology).

Another significant advantage is the internationality of such a semantic tagging approach, based on the fact that Wikipedia relates concepts across languages. For example, a user in Bahrain could ask for pictures tagged with an Arabic word, and would also receive pictures originally tagged with equivalent Chinese words. To some extent, semantically interlinking articles across different languages would even allow to retrieve translations of single words—especially when including Wiktionary¹², a MediaWiki-based dictionary that already exists for various languages.

The wealth of data provided by Semantic Wikipedia can also be used in research and development. A resource of test data for semantically enabled applications immediately comes to mind, but social research and related areas can also profit. For example, comparing the semantic data, one can highlight differing conceptualisations in communities of different languages.

For the SEKT project we expect Semantic Wikipedia to offer a wealth of semantically annotated text. This can be both used to test the Knowledge Discovery and Human Language Technology tools that have been developed in the scope of the project. We also expect the exported database of Semantic Wikipedia to be a stress test for any triple store and inference engine. Finally we plan to install a Semantic MediaWiki system within the Siemens case study in order to evaluate the introduction of the system in a corporate setting.

7.6 Conclusions

We have shown how to create a integrated wiki extension to allow end users to create machine-processable knowledge using semantic technologies. On the user side, our primary change is the introduction of *typed links* and *attributes*, by means of a slight syntactic extension of the wiki source. For further processing, this knowledge is conveniently represented in RDF-based format.

We presented the system architecture underlying our actual implementation of these

¹²<http://wiktionary.org>

ideas, and discussed how it is able to meet the high requirements on usability and scalability we are faced with. The outcome of these considerations is a working implementation which hides complicated technical details behind a mostly self-explaining user interface.

We have demonstrated that the system provides many immediate benefits to the wiki's users, such that an extensive knowledge base might be built up very quickly. The emerging pool of machine accessible data bears great opportunities for developers of semantic technologies who seek to evaluate and employ their tools in a practical setting. In this way, the Semantic MediaWiki can become a platform for technology transfer that is beneficial both to researchers and a large number of users worldwide, and that really makes semantic technologies part of the daily usage of the World Wide Web.

Chapter 8

Outlook

Here we present an outlook of how we expect the methodology to evolve during the next year of the SEKT project. The case studies will provide input and experiences towards the application of the methodology and the usage of the tools developed in SEKT based on the three SEKT technologies.

We therefore take a look at the three case studies.

8.1 Siemens Case Study

The objective of this case study is to investigate and verify how semantically enabled technologies can improve the productivity of IT and business consultants. It will elaborate which added values can be created and look for new forms of accessing, handling and utilising content in the IT and business services industry. Complementary to the other two case studies, it will focus on how to stipulate the emergence and creation of new knowledge and its capture, thereby providing important input for the further shaping of SEKT in subsequent phases.

A key point of the case actually is the integration into the business processes. This is a particular distinction compared to the other two cases studies. We expect the Siemens case study to feed back experience on this aspects of the DILIGENT methodology.

Also the Siemens partner has integrated several SEKT technologies into the SIP platform, and plan to integrate further. The usage of SIP is expected to be an interesting addition to the range of experiences we will harvest within SEKT.

8.2 Legal case study

The legal case study has provided us with the strongest feedback during the second year of SEKT. In the third year we expect the OPJK to evolve because of three new aspects, and they will give feedback to both the DILIGENT methodology and the applied SEKT techniques:

1. In the legal case study, the ranking and selection of the presented question/answer pair will be based on the ontology. Evolving the ontology to reach a higher precision/recall rate is one expected course.
2. Missing questions and answers will be fed back to the Iuriservice system. The ontology needs to adapt to these new entries and still retain a constantly high reliability.
3. More questions will be analysed and the different ontological sub-domains added to OPJK.

Moreover, the development of the second part of the prototype, related to the retrieval of relevant judgements (so they exist) for the retrieved question/answer pair, will produce at least an ontology for the judgements. The creation and development of this ontology will also apply and get support from the DILIGENT methodology and other SEKT techniques, as mapping technologies for ontologies (OPJK and judgement) will be required.

8.3 Digital Library Case Study

The digital library will use topics for the organization and browsing of the items in the library. The topics are being drawn from a topic ontology. We can describe the change management process of the topic ontology with DILIGENT. After the initial **build** of the ontology, it is delivered to the users. The user profile is meant to be highly adaptive, actively - with the user creating information spaces and thus expanding the ontology - and passively - with the system logging and analysing the users actions, and thus further adapting the user profile with this usage driven changes [HV04]. This step represents the **local adaptation**. **Analysing** these adaptations by the board responsible for the quality of the Digital Library will show up the deficits of the current ontology and allow to **revise** it properly, in order to achieve a higher usability and a better performance by changing the topic relations, especially the hierarchy, and by keeping the information spaces close to the current interests of the users and easy accessible. This way the board has constant feedback on the interests of the users, who **update** their data accordingly.

In this case study the argumentation procedure will actually be replaced by SEKT technologies capturing the user's need and delivering them for analysis in a machine readable way. In the following we outline some scenarios:

1. The DILIGENT process will apply to evolving the BT Digital Library topic ontology. Especially inside the “Jot” knowledge sharing application [SEK05a] the DILIGENT process will be followed by the users to introduce their own topics to describe the ‘pages’ they wish to ‘Jot’ (the **local adaptation** step). Furthermore, people copying other people’s ‘jottings’ may well use different descriptors (very likely) to describe content, i.e. descriptors that are more ‘meaningful’ to the individual.
2. Mapping technologies could be used to reconcile different users jottings. The BT digital library topic ontology could then be extended accordingly. For example, a user may describe a search engine document in terms of a project, rather than in terms of a technology; a relationship could then be introduced between the topic from a technology perspective and the use of that technology in a project (e.g. link the topic to an Intranet page describing a project).
3. The initial topic ontology will be closely aligned with the ‘descriptors’ (controlled indexing terms) of the Inspec and ABI thesaurus. Information space queries will initially be defined using these descriptors. It is very likely that as the information spaces evolve (public and private); using free-text terms/phrases that are not defined in the thesauruses. The DILIGENT process will be used to identify ‘new terms’ and put these forward for inclusion in the topic ontology.
4. The topic ontology will also evolve through use. New topics will be introduced into the ontology. Subsequent updates to the ABI and Inspec thesauruses may include some of the ‘new topics’ that were introduced into the topic ontology. There will be a need to merge the evolving BT topic ontology with the descriptors in the updated thesauruses. The DILIGENT processes will be applied here as well.

As a user browses her intranet or the internet, the knowledge sharing tool will classify the current page against the Topic hierarchy, and automatically fire a query to fetch other pages in related Topics, as defined by the current domain of interest. Furthermore, as the user browses web pages, the knowledge sharing tool will give priority to web pages the user’s team members have chosen to share. By following the related pages, the user can make and amend comments or read comments left by their colleagues about those web pages. Importantly, the user can share pages to herself, which makes the software useful even if no contributions are made by members of the wider community. The user is able to view a ranked list of pages associated with the currently viewed Topic: the list is built from references pages with the Library, pages visited by the user and pages shared by other users. A greater weighting is given to pages shared by close neighbours in the user’s social network.

The knowledge sharing tool allows web pages to be shared to other members of the user’s immediate Team, looser communities of which they are a member or more widely. By sharing a web page, the user is also given the opportunity to state that the page belongs to their own local Topic, in parallel to the official Topic set, which allows a community to

have its own vocabulary. However, the tool places greater emphasis on pages shared by team members. In this way a shared page is viewed through both the social context and technical context: a page is classified as belonging to “Internet/Semantic Web” as well as being seen as important by colleagues. As more people contribute pages with their own Topic names and associations, an evolving community Topic hierarchy will develop.

Chapter 9

Conclusions

In the SEKT project we developed a methodology for the development of ontologies and the application of semantic technologies into enterprises in a goal-driven and process-oriented way [STV⁺05]. The methodology takes special care at how to integrate together the functionality of the three core technologies being developed in SEKT. The final version of the methodology will give detailed insight on technical aspects, but also cover organisational effects –e.g. what barriers could block the introduction of SEKT technologies in companies.

With this deliverable we provide a collection of best practices and lessons learned during the first two years of the SEKT project. Some of this experiences led to the development of the tools described in this deliverable, the DILIGENT tool, geared toward the distributed engineering of ontologies, and a semantic wiki, especially useful for the population of ontologies.

This is a list of the lessons learned described in this report. We point to the sections where we outline each lesson in more detail.

- the data must be available in sufficiently large quantities before the knowledge discovery technologies can take place (2.2).
- collecting the data can be expensive and legally problematic, and it is not guaranteed that the quality will be sufficient (2.2).
- preparing the data involves both knowledge discovery experts, and domain experts. In the case of knowledge discovery from text, some experience with natural language processing is required (2.2).
- when working with text, one has to mind the semantics of individual words: some words are relevant, others irrelevant; some words have multiple meanings, some meanings have multiple words. The words have to be mapped into meaningful primitive concepts (2.3).

- when working with concepts, one has to mind the association between concepts and words. This is achieved by examining the overall classification performance, and by investigating the causes of classification mistakes (2.3).
- if the goal is effective automatic classification, the ontology must be designed in a way that allows this. The classifier can provide feedback to the domain experts and knowledge engineers. When a classifier is having problems with an issue, a human being might have problems too (2.3).
- in order to enable easy integration of HLT tools within end-user applications, a comprehensive HLT infrastructure is required, which addresses issues such as diverse document formats, multiple languages, scalability, portability, and robustness (3).
- Multilinguality is particularly important, especially for European companies which supply products to diverse markets. This is especially true for components like lemmatizers and stemmers, which need to be rewritten by experts for other languages as they can't be merely translated (3).
- given a powerful HLT infrastructure on one hand, and sufficient communication between domain experts and application builders (e.g., ontology engineers, HLT experts), existing tools or new tools can be built or adapted for specific needs with relatively little effort (3).
- adhering to a proper development cycle for HLT-based applications is crucial: starting from data collection, component modification, and finally quantitative evaluation on the data to determine how effective the HLT technology is in addressing the user requirements (3).
- making modules instead of one big monolithic ontology enables reuse more easily (4.1).
- light-weight ontologies make for an easier reuse by non-experts, as they don't have to buy possible logical consequences they maybe don't realize. This also lends to an easier understanding than heavy weight ontologies provide (4.1).
- reusing an upper level ontology can ease the process of creating one from scratch – if they indeed are easily reusable (4.1).
- simple names, labels and sufficient comments of ontological entities are important. It is easier to reuse *Happening* and *Object* than *Endurant* and *Perdurant*, respectively, especially when they are well-documented (4.1).
- a well-defined process to feed back change requests lead to less frustration on the (re)users' side (4.1).

- clustering ontological entities by topic under a common concept eases up the engineering and understanding of an ontology (4.1).
- in order to manage ontologies more easily, it often can be fruitful not to exploit their whole expressive power, but only as much as needed for the task at hand (4.2).
- as we can see in the case studies, even “a little semantics can go a long way”. We don’t have to use the full expressive power of a language like OWL DL in order to benefit from ontologies (4.2).
- competency questions should be collected from the actual users, and not created by knowledge engineers or theoreticians (5.2).
- sometimes it is hard to speak to the expected users, because they may have a different background, live in a different culture, basically speak a different language. But even though it is hard – it is worthwhile (5.2).
- professional knowledge is different than theoretical knowledge. When building ontologies that are to model professional or tacit knowledge, stick to the collected competency questions and ignore other sources (5.2).
- Certain types of ontologies, especially those related to professional knowledge, have to be build by domain experts and ontology engineers together. So it is important that they have tools (distributed environment) to communicate changes in the ontology. And as domain experts have to be involved in the process, tools have to be domain expert friendly – not just for nerds (5.2).
- off the shelf tools help in the first validation of research ideas, but have to be replaced by dedicated tools for real application (5.3).
- a strong integration between ontology tools is crucial. This will help users to come up much more easily with new and innovative ideas to use given tools (5.3).
- visualisation is crucial. The users have gone great lengths to provide a visualisation, even if it meant a lot of manual and repetitive work (5.3).
- it is important to follow a methodology when building an ontology – it both clarifies the objectives and provides a list of things to do (5.4).
- when building an ontology, tracing the argumentation in some way speeds it up considerably (5.4).
- it is important to find good evaluation measures for the ontology once build (5.4).

Several lessons learned still point to open issues – not only on the technical side. How to manage the possible complexity of ontologies? How to solve social issues? How to

address the problem of the availability of data? In this deliverable we collected these issues.

In the next year, the SEKT case studies will provide further input towards the refinement of the ontology. We will continue to collect the lessons learned and produce a final description of the SEKT methodology.

Appendix

Appendix A

Description of EvA screens

This part of the document describes the screens as designed in the low fidelity mock-up. This mock-up is done in a Microsoft PowerPoint presentation.

If a screen element or functionality is marked with * it is mandatory and expects data from the user.

A.1 Screen layout conventions of the eclipse platform



































































create, new		compare		forward		jar		plugin	
save		debug		backward		WAR		extension	
cut		run, execute		previous		EAR		extension point	
copy		import		next		window		thread	
paste		export		project		perspective		process	
add		play, resume		open project		property sheet		mapping	
remove		suspend		folder		table		error	
delete		terminate		open folder		database		warning	
erase, clear		stop		file		repository		alert	
search		undo		library		class		conflict	
find		redo		package		interface		public	
help		refresh		session bean		attribute		protected	
edit		filter		server		element		private	
								default	

Figure A.1: Screen buttons as recommended by [EHLPO4]

Other conventions:

- If not enough screen space is available to display a whole name or label, a tool tip text is used to provide the whole name.

- Button “K” is used to commit a change operation on the applications data model. It returns back to the screen where the user was before. The changes the user has performed are visualized
- Button “Cancel” is used to return to the previous screen without any changes made.
- A click into the white space of the visualization view will deselect all selected modelling primitives.

A.2 Screen design

- Project configuration:
 - Configuration for all the Ontology constructs that can be used in a project. This makes sense because it would hide unused functionality from the user (e.g. hasValue, or unionOf, etc.)
 - Ontology Editing mode (ontology changes without argumentation) vs. Argumentation mode (Ontology changes can only be done if an argumentation is provided.)
 - Configuration of voting mechanism. E.g. how many team members should agree on a concept in democratic mode or if the ontology owner decides, which concepts are going to be used.
- Personal configuration
 - Configuration of notification mechanism: The user can decide if she wants to be notified via e-mail, the internal task list of the application or instant messenger.
- Wizard for creating a new project locally and on the ontology engineering server.
- Some detailed features are missing, which are described in the scenarios. E.g. it should be possible to add more than one person as the creator of an idea, issue, or argument.

A.3 General Design Principals

Argumentation An argumentation in this application can be about

- an idea in form of a
 - concept

- relation
 - instance
 - sub-relation relation
- an idea set
- an issue in form of an
 - issue
 - elaboration

If one would want to discuss about more than one modelling primitive, that reaches across issues, a new issue needs to be raised. This new issue can then be discussed.

Ontology change operation An idea, which is always a creation, edit or deletion of a new modelling primitive, is a possible ontology change operation. As soon as users have voted on an idea or the ontology owner decides the ontology is being changed. Therefore the voting is the actual ontology change operation.

Voting mechanism The voting can be done in two ways. Either a moderator decides if an issue or idea is agreed or the software determines that an issue or idea is agreed because a defined percentage of users has agreed (e.g. 50%).

The voting is always independent from the discussion and the user providing a pro or contra argument. The reason is that a user might not have formed her opinion yet as the issue is still under discussion.

It needs to be clear to the user that in the democratic mode her vote influences in the end the actual ontology change operation. If the threshold for agreeing or disagreeing over a concept has been reached the ontology is actually changed. The issue is then agreed and closed. If a user or a software agent has found an error or a user wants to reopen the discussion, she has to raise a new issue.

The voting mechanism should be carefully encapsulated when implementing the application. The actual voting has not been tested in a real environment, but only in thought experiments. It might need change.

Folder structure The folder structure of the source view is synchronized with the issue view and in the task view if tasks are generated automatically from all (imported) sources. The task view can have additional folders, which group general tasks.

Drag and Drop All fields that expect input from something that is visible on the screen, can be filled via drag and drop. For example it is possible to drag modelling primitives over a modelling primitives list box in the new/edit issue dialog window (see section A.5.11). This will drop (add) the “dragged” modelling primitive on the list.

A.4 Requirements not regarded

In the prototype some detailed manipulation mechanism of modelling primitives are not included in the design:

- Cardinality
- Transitivity of relations
- Symmetry of relations
- unionOf
- hasValue
- sameAs
- Equivalents
- differentFrom
- Disjoint

The selection of possible data types is incomplete. In this prototype only “string”, “number”, “integer”, and “boolean” are used.

A.5 Description of screens

A.5.1 Main argumentation window

Type: Window

Name: Argumentation window (see Fig. A.2)

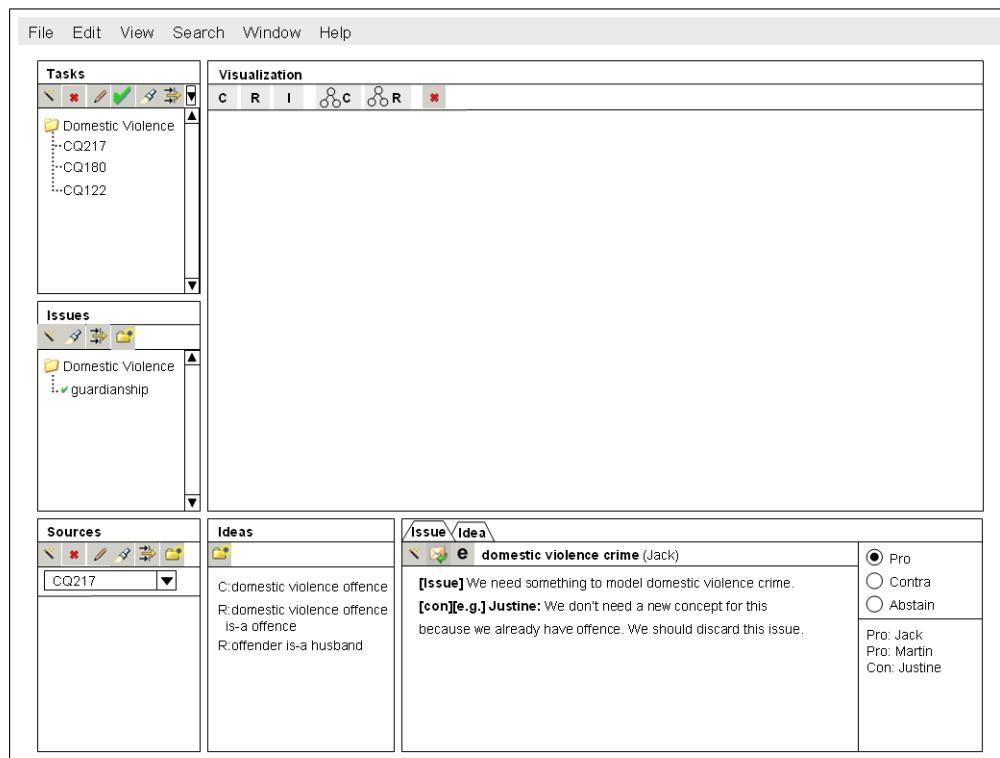


Figure A.2: Main argumentation window

Menus: The menu structure has not been tested. Generally the menu structure reflects the eclipse standards as documented in the Eclipse Style Guide [EHLPO4].

- File
 - New ontology
 - New project
 - _____
 - Open ontology
 - Open project
 - _____
 - Save ontology
 - Save ontology as
 - Save project
 - _____
 - Import
 - * Import ontology

- * Import source
 - Export
 - * Export ontology to RDF
 - * ... (continue with different formats)
 - _____
 - Print
 - Exit
- Edit
 - Undo
 - Redo
 - _____
 - Cut
 - Copy
 - Paste
 - _____
 - Delete
 - Select all
 - _____
 - Find / Search
- Modelling
 - Switch to ontology editing mode (is only shown to project owners and administrators)
 - New concept
 - New subconcept
 - _____
 - New property
 - New subproperty
 - _____
 - New instance
 - _____
 - New idea set
 - _____

- Edit
 - _____
 - Delete
 - _____
 - Vote (screen has not been designed, voting can be seen on screen A.5.1)
- Project
 - New source
 - New task
 - _____
 - Create tasks for sources (creates one task for every source asking the user to create issues regarding this source; a better labelling might be possible)
 - Create review tasks (creates a task for everybody else to review the ideas connected to an issue)
 - _____
 - New issue
 - New elaboration
 - _____
 - Manage persons (screen has not been designed)
 - Voting mode
 - Manage notifications
- Window
 - Reset visualization
 - _____
 - ' as in eclipse (Eclipse User Interface Guidelines Version 2.1, p 2)
- Help
 - Start tutorial
 - _____
 - ' as in eclipse (Eclipse User Interface Guidelines Version 2.1, p 2)

General remarks: If a task is selected the sources are filtered so that only the related sources are shown. If an issue is selected only the sources are selected. Either the task or issue are filtering the sources depending on which one was selected last.

View tasks:

- Button “new” opens a new dialog (see A.5.8) for creating a new task.
- Button “delete” opens a dialog: “Do you want to delete the task(s) [name of the task]?” with the two buttons “ok” “cancel”.
- Button “edit” opens a new dialog (see A.5.8) similar to the “new” dialog. The data of the selected task is filled into the fields and can be edited by the user.
- Button “mark as done” marks the selected tasks as done.
- Button “search” opens a dialog (see A.5.10). In this dialog the user enters the text string she is searching for. With the click on “ok” the first finding is highlighted. A further click on “ok” will highlight the next finding. When all findings have been displayed a dialog message will be displayed saying “All issues have been searched. Search continues at the beginning” with an “ok” Button. The search dialog remains open until the user explicitly closes it.
- Button “filter” opens a new dialog for filtering tasks (see A.5.9).
- Button new folder is not visible but, accessible via the drop-down-box on the right hand side at the top of the view. A click of on the button creates a new folder for organizing tasks.
- The status (open, postponed, done) of the task is shown by small icons in front of the task name.
- Tasks can be group recursively by folders.
- By selecting a task the user filters the sources. If a task is selected only those sources are shown, which have been assigned to a task. The first source is selected and shown on the screen.
- Context Menu (opens on right mouse click on task):
 - Show task
 - Edit task
 - ———
 - Mark done
 - Postpone task
 - ———
 - Delete task

View issues:

- Button “new” opens a new dialog (see A.5.11) for creating a new issue.
- Button “search” opens a dialog (see A.5.11). In this dialog the user enters the text string she is searching for. With the click on “ok” the first finding is highlighted. A further click on “ok” will highlight the next finding. When all findings have been displayed a dialog message will be displayed saying “All issues have been searched. Search continues at the beginning”. The search dialog remains open until the user explicitly closes it.
- Button “filter” opens a new dialog for filtering issues. Options are to filter issues by status within folders. If a folder contains only items, which are not shown by the filter the folder is also hidden.
- The status (under discussion, postponed, agreed, discarded) of issues is shown by small icons in front of the issue name.
- Only one issue can be selected at a time because only one argumentation in combination to the issue can be visualized on the screen.
- Only the issues of the active task are shown in the issue view.
- Context Menu (opens on right mouse click on task):
 - Show
 - Edit
 - Create elaboration
 - ———
 - New argument
 - Vote
 - Create review tasks
 - Postpone issue
 - ———
 - Discard issue

Sources view:

- Button “new” opens new dialog (see A.5.12) for creating a new source
- Button “delete” opens a dialog: “Do you want to delete the source(s) XYZ?” with the two buttons “ok” “cancel”.

- Button “edit” opens a new dialog (see A.5.12) similar to the “new” dialog. The data of the selected source is filled into the fields and can be edited by the user.
- Search opens a dialog (see A.5.10). In this dialog the user enters a text string she is searching for. With the click on “ok” the first finding is highlighted. A further click on “ok” will highlight the next finding. When all findings have been displayed a dialog message will be displayed saying “All sources have been searched. Search continues at the beginning. The search dialog remains open until the user explicitly closes it.
- Button “filter” opens a new dialog for filtering the displayed sources. The default filter is that only the sources related to the active issue or the active task are shown. The other options are to view all sources or to filter by source type (see A.5.9)

Argumentation view: In this view the argumentation regarding the selected issue is shown. An argument can be on the issue or an idea within the selected issue (see also A.3 and A.4).

- Status is shown. It can be under discussion, postponed, agreed, or discarded. An issue is agreed when all ideas have been voted in favour of. It is discarded if the voting result is against the issue (In democratic mode: a certain percentage of people has voted against the issue).
- Button agree/disagree: are used for voting. If clicked either pro or con and the name of the voter are shown below.
- Button “new” creates a new argument that is not related to another argument already on the screen.
- Button “respond” creates an argument that is a response to the currently selected argument.
- Button “elaboration” (e) creates an elaboration (or sub-issue) to an issue. It is displayed hierarchically in the issue view.
- The first argument of an issue, idea, or elaboration is always selected, if an issue is has been selected by a user.
- Naming conventions in the argumentation view are used for displaying issues and arguments in the argumentation view. (formal syntax: | = or; = optional, CAPITAL = a variable name).
 - “[issue] ISSUE” In front of an issue is always the label “issue” displayed (e.g. [issue] We should model marriage...).

- “Argument: [pro | con] [e.g.] + PERSON: ARGUMENT” To each argument the information if the argument is pro or con is added. If the argument is an example “[e.g.]” is displayed. Additionally also the name of the person providing the argument is displayed (e.g. Argument: [pro][e.g.] Justine: I am in favour of...)
- Tabs:
 - * The tabs are not named dynamically. There are two tabs that have the label issue and idea.
- Heading in the argumentation view:
 - * “C: CONCEPT_NAME” is used for describing a concept.
 - * “R: CONCEPT_NAME1 RELATION_NAME CONCEPT_NAME2”
The syntax above is used to label a relation. A sub relation has as relation name “is-a”. If a concept is a sub-concept of the root concept only the concept name is displayed.
 - * “I: INSTANCE_NAME” is used for labelling instances.



Ideas	Issue \ Idea	
 C:domestic violence offence R:domestic violence offence is-a offence R:offender is-a husband	 domestic violence crime (Jack) [Issue] We need something to model domestic violence crime. [con][e.g.] Justine: We don't need a new concept for this because we already have offence. We should discard this issue.	<input checked="" type="radio"/> Pro <input type="radio"/> Contra <input type="radio"/> Abstain Pro: Jack Pro: Martin Con: Justine

Figure A.3: Ideas and argumentation view with active voting mechanism. In the idea view all corresponding ideas (expressed as modelling primitives) are shown. In the argumentation view the arguments are shown. The voting can be done via the radio buttons on the right hand side. If the issue is postponed, discarded or already agreed upon the voting mechanism is disabled. The names of the team members who have already voted are shown below.

Idea view: In this view the different idea sets and ideas on an issue are shown.

- Principally the ideas of the first idea set are visualized. If the user chose ideas as being contradictory, when creating or editing a new idea set then these ideas are hidden in the visualization window. They are shown again when the user selects the first idea set (the default idea set) or any other idea set where the ideas are not chosen as being contradictory and therefore hidden.
- The status of each idea is visualized in the idea view. In front of each idea the same symbols are used as for the visualization of the status of issues. The status can be under discussion, postponed, agreed, or discarded.

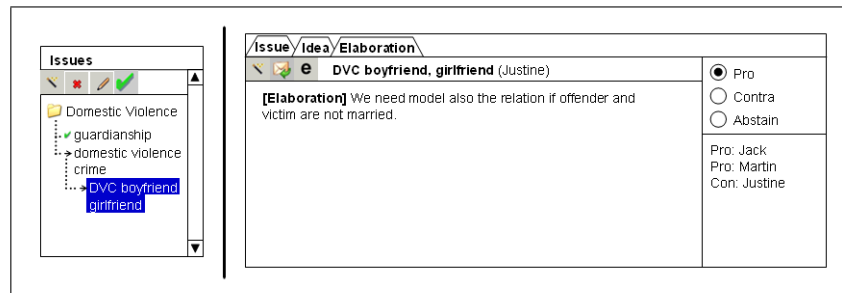


Figure A.4: Issue view and argumentation view with an elaboration. In the case of an elaboration a third tab is added, which holds the arguments for the elaboration. In the issues view an elaboration is also added.

Remarks on the visual editor: Generally the visual editor is out of the scope of this document. The interaction model of the KAON OI modeller is used. Here some remarks are collected how the visual editor and the argumentation interaction design are tied to together. Additionally some comments are made, which might improve the interaction with the KAON IO modeller as visual editor.

- There are two buttons for creating a new concept. One will create a new concept that is placed on the same level of the hierarchy as the node that has been selected and one the will create a sub-concept of the selected concept (the one with the hierarchical structure). If no concept is selected both buttons will create a sub concept of the root concept.
- Visualization of voting: If a modelling primitive is new, it is visualized with traffic lights red and green. If no votes have been given yet, the red and the green light are shown with opacity of 25% on white ground. If the majority of the votes is in favour of the modelling primitive, the status of the green traffic light changes to 100% opacity. In the opposite case the status of the red traffic light changes to 100% opacity. If the majority changes the status is switched. Important is that the opacity is also changed so that color blind persons are also able to recognize the difference. The shape of the traffic light is not fixed and can be changed so that the voting is bigger. There is no interaction on the traffic light. If the user wants to vote she has to select (with a click) the modelling primitive. Then the argumentation of the idea is shown in the argumentation view and the user can vote.
- If a single idea has been selected, the connected modelling primitive is selected, too.
- Visualization of idea sets: Whenever an idea set is selected the modelling primitives belonging to that idea set are shown with 100% opacity. All other shape of the modelling primitives are shown with opacity of 55%, but the connecting lines between the modelling primitives always have an opacity of 100% (see figure A.5).

- If a single issue had been selected, the original (first) idea set is selected and the concepts are shown as defined above.
- Context Menu (opens on right mouse click on task):
 - Edit
 - ———
 - New subconcept
 - New property
 - ———
 - Show/hide subconcepts
 - Show/hide superconcepts
 - Show/hide properties
 - Show/hide instances
 - ———
 - Show/hide issue
 - Show/hide argumentation
 - ———
 - Delete

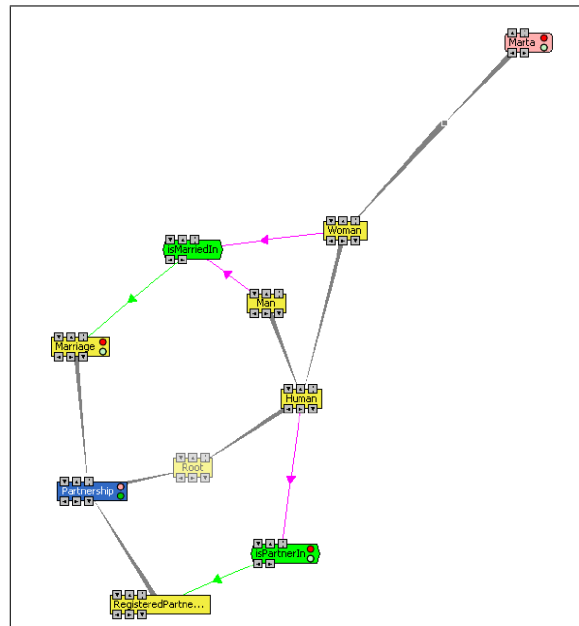


Figure A.5: Example for the visualization of an ontology with voting.

A.5.2 Dialog “Create new concept”

Type: Dialog window

Name: Create new concept (see Fig. A.6)

Figure A.6: Dialog for creation a new concept.

With this dialog window the user creates a new concepts and with that new concepts he creates a new idea. This means she needs to provide an argument if the application is in argumentation mode. If the entry for a cell is required a * appears in front of the cell (this is valid for all dialog descriptions in this chapter).

Functionality:

- * Label text field expects as input the label of the concepts. If the URI field is empty the entry from the label text field is taken to create a new URI for the concept.
- Synonyms text field expects the entry of any synonyms of the label.
- Comment text field can be used for additional comments and documentation purposes.
- * URI text field is a combined drop-down-box and text field (comparable to the address bar in a web browser). With the help of the drop-down-box namespaces can be selected for the easy creation of a concept’s URI. To make the URI valid a label needs to be added.
- * Argumentation text field expects an argument why this idea has been introduced. The argument is always classified as “pro”.
- Example check box needs to be ticked if the user provided an example and not an evaluation & justification argument.

Error messages:

- “The given URI is already in use for another concept. Please choose a different URI. Please also consider a different label for the concept.”

A.5.3 Dialog “Select sources”

Type: Dialog window

Name: Select sources(see Fig. A.7)

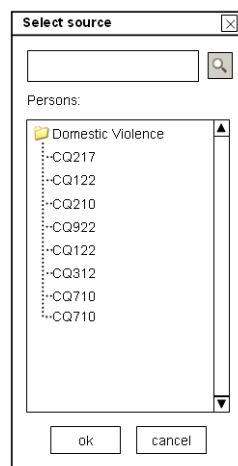


Figure A.7: Dialog for selecting sources.

Functionality:

- Search/find: Text field to enter a source name, which is then selected. If the search string exists more than once the first occurrence is highlighted. A further click on find will show the next occurrence. When all names are found a message box appears indicating that all persons have been searched. If the user clicks again the first finding will be shown again.
- In the source selector the user can select one or more source (by holding down the “ctrl”-key).

The dialogs “A.5.4”, “A.5.5”, and “A.5.6” have a similar functionality. The screen design is adapted a bit to the different purposes.

A.5.4 Dialog “Select issue”

Type: Dialog window

Name: Select issue (see Fig. A.8)



Figure A.8: Dialog for selecting issues.

In this dialog the user selects one or more issues from all available issues.

Screen elements / Functionality:

- See A.5.3.

A.5.5 Dialog “Select person”

Type: Dialog window

Name: Select person (see Fig. A.9)

In this dialog window the user is able to select a team member of the current project.

Screen elements / Functionality:

- See A.5.3.

A.5.6 Dialog “Select concept”

Type: Dialog window

Name: Select concept (see Fig. A.10)

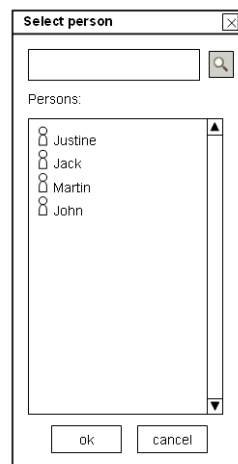


Figure A.9: Dialog for selecting team members.

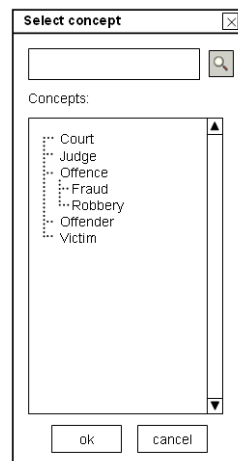


Figure A.10: Dialog for selecting concepts.

In this dialog window the user is able to select a concept from the concept hierarchy.

Screen elements / Functionality:

- See A.5.3.

A.5.7 Dialog “Select instance”

Type: Dialog window

Name: Select instance (see Fig. A.11)

This window is for selecting instances.

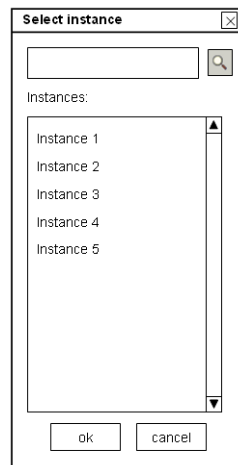


Figure A.11: Dialog for selecting instances.

Screen elements / Functionality:

- See A.5.3.

A.5.8 Dialog “Task X”

Type: Dialog window

Name: Task X (see Fig. A.12)

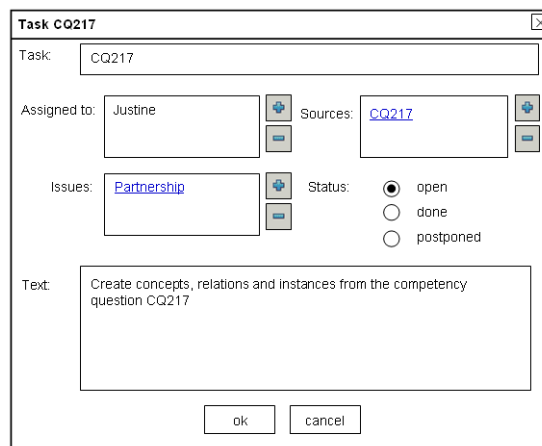


Figure A.12: Dialog shows the view on one particular task.

This dialog window represents task X. If a new task is created only the source selector is filled with one or more sources in case a source was selected when the task was created.

Functionality:

- * Task text field expects the name of the task.
- Assigned to selector: with a click on + a new dialog window opens as in A.5.5 for adding persons. With a click of - all selected persons are being removed.
- Sources selector: With a click on + a new dialog window open as in A.5.3 for adding sources. With a click of - all selected sources are being removed.
- Issues selector: With a click on + a new dialog window opens as in A.5.4 for adding issues. With a click of - all selected issues are being removed.
- Status selector: Only one of the three selectors can be active at a time. It defines the status of the task which is also visualized in the task view. Default value is the status “open”
- Text: Expects any additionally free text comments or description of the task.

A.5.9 Dialog “Filter tasks”

Type: Dialog window

Name: Filter tasks (see Fig. A.13)

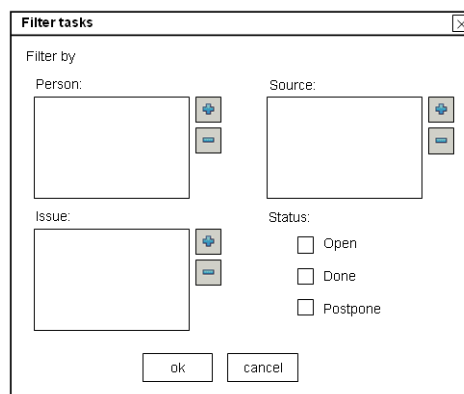


Figure A.13: Filter tasks dialog.

This window allows to filter, which tasks are to be displayed in the application. The filter can be either person, status, source, or issue.

Functionality:

- Persons are filtered by the list of the team members. The default is that only the current user's tasks are displayed. Persons can be added using a new dialog (see A.5.5) and removed with the remove button.
- Sources are filter by opening a new dialog (see A.5.5). Single sources as well as source folders (as a set of sources) can be selected. When the user confirms his action only tasks in relation to the selected sources are shown in the task view.
- The status is filter using checkboxes. Only tasks with state that are selected are shown when the action is confirmed (click on "ok", enter, etc.)
- Issues are filtered by selecting the issues or folders of issues to be displayed via a new dialog window (see A.5.4). They can be removed by using the remove button. If the user confirms the action, only tasks that are related to the selected issues and folders of issues are displayed.

A.5.10 Dialog "Search"

Type: Dialog window

Name: Search (see Fig. A.14)

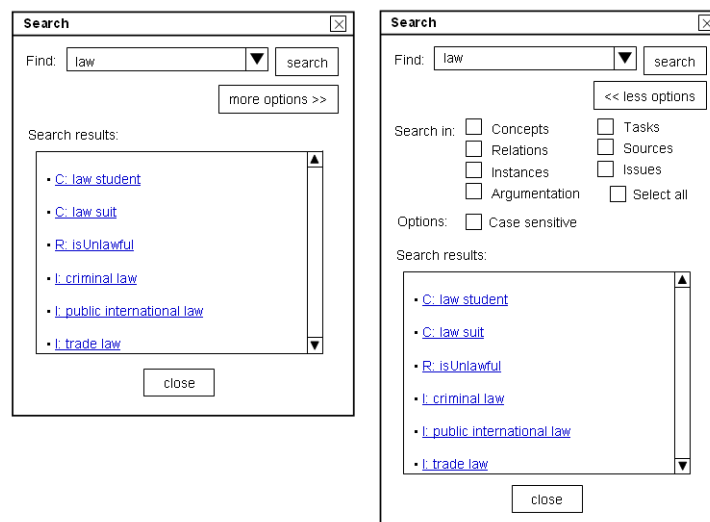


Figure A.14: Search Dialog with and without advanced options.

This window allows searching across all textual data in the application. The dialog window is floating over main window. It is not closed by the selection of a search result, but only by the button "close".

Functionality:

- * Find text field and drop-down-box: Expects a text entry or a selection of text by using the drop-down-box functionality.
- Button “more options >>” “<< less options” Shows or hides the Search in check boxes and the case sensitive check box. The application remembers the last status of the box if the window is closed.
- Search in check boxes: By default all checkboxes are selected. If a checkbox is selected the corresponding data is being searched. The check box “select all” selects all checkboxes in the “search in” context if not all check boxes are selected. If the last checkbox of all other check boxes is selected “select all” is also selected. All check boxes in the “search in” context are selected and the user deselects the select all checkbox all other check boxes are also deselected.
- Case sensitive check box: If the checkbox is selected the search is case sensitive.
- Search results list box: If the user clicks on one of the search results, the result is highlighted and visualized as if she had clicked on the item in the argumentation window. The dialog window will keep the focus as long as the user does not perform any action in the argumentation window.
- Button “close” closes the window. The state of the options is remember (options hidden or visible).

A.5.11 Dialog “New issue”

Type: Dialog window

Name: New issue (see Fig. A.15)

In this window the user can create a new issue.

- * Text field with issue name.
- The status options can be set manually to open or postponed. When the user creates a new task the default setting is “open”.
- The modelling primitives view allows to list the user all modelling primitives, which might be affected by the issue. The field can also be empty.
- The sources view allows to list the sources, which led to raising the issue at hand. The field can as well be empty.
- Text field “text” allows a more depth description of the issue at hand.

Figure A.15: Dialog new issue.

A.5.12 Dialog “New source”

Type: Dialog window

Name: New source (see Fig. A.16)

Figure A.16: Dialog for creating a new source.

In this window the user can create a new source or a link to a new source.

Functionality:

- Text field “source label” is the label shown in the main screen for selecting a source. This implies that the labels entered by the user should be relatively small. The text field is short on the screen, but it can contain texts longer than the number of characters the text field displays.

- External source is for creating a URI either as Hyperlink or URI or as file. When file is selected the browse button is shown for browsing to a file.
- Text field “text” allows to enter a short text as source or as description of a source.

A.5.13 Dialog “Watchlist for issues”

Type: Dialog window

Name: Watchlist for issues (see Fig. A.17)

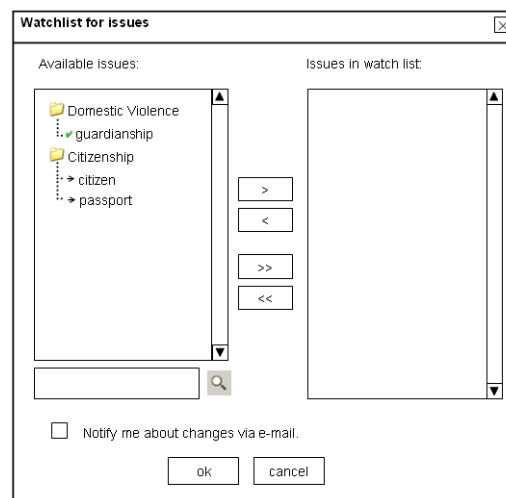


Figure A.17: Dialog for creating watchlists for issues.

This window allows creating a watch list for issues. If a changes occurs to issues in this watch list the user will get a new task remembering him to review an issue if it changed. Additionally he can set a preference for sending a notification via e-mail.

Functionality:

- Button “>” adds a selected issue to the watch list.
- Button “<” removes a selected issue from the watch list
- Button “>>” adds all issues to the watch list.
- Button “<<” removes all issues from the watch list.
- Button find searches for the first item on the list of issues that matches the entry in the text field.

A.5.14 Dialog “New argument”

Type: Dialog window

Name: New Argument (see Fig. A.18)

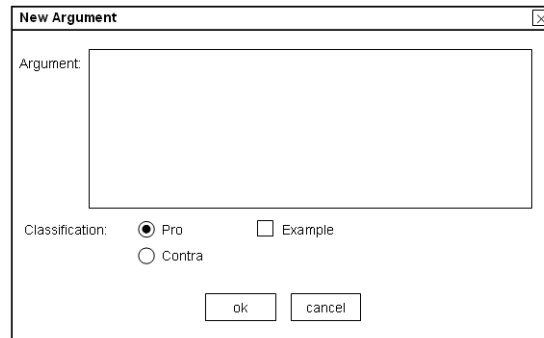


Figure A.18: Dialog for creating a new argument.

In this window the user can create a new argument.

Functionality:

- Argument text field is where the user enters the argument
- Pro/Contra: The selection elements allow only the Boolean setting: “Pro” or “Contra”
- Example check box is stating that an argument is either an example or a counter example depending on the selection elements. If “pro” is selected it is an example, if “contra” is selected it is a counter example.

Additional information to be stored with a new argument:

- User names of the persons creating the argument. This might be in fact more than one person.

A.5.15 Dialog “New elaboration”

Type: Dialog window

Name: New Elaboration (see Fig. A.19)

With this dialog window the user can express an elaboration on an issue.

Screen elements:

Figure A.19:

- Issue displays in a text field the issue that the new elaboration is about. The text field is not editable, which is shown by a visual cue. Functionality:
- Elaboration text field allows naming the elaboration.
- For all other fields see new issue A.5.11.

A.5.16 Dialog “New relation”

Type: Dialog window

Name: New Relation (see Fig. A.20)

This dialog window allows the user to create a new relation between two or more concepts.

Screen elements / Functionality:

- * Label text field expects as input the label of the relation. If the URI field is empty the entry from the label text field is taken to create a new URI for the relation.
- Comment text field can be used for additional comments and documentation purposes.
- * URI text field is a combined drop-down-box and text field (comparable to the address bar in a web browser).. With the help of the drop-down-box namespaces can be selected for the easy creation of a concept’s URI. To make the URI valid, a label needs to be added.

The 'New Relation' dialog box is a standard Windows-style window with a title bar and a close button. It contains the following elements:

- Label:** A text input field.
- Comment:** A text input field.
- URI:** A text input field with a dropdown arrow on the right.
- Domain:** A large rectangular area for selecting concepts, with a '+' button to add and a '-' button to remove.
- Range:** A large rectangular area for selecting concepts, with a '+' button to add and a '-' button to remove.
- Argumentation:** A large text area for providing an argument.
- Example:** A checkbox labeled 'Example'.
- Buttons:** 'ok' and 'cancel' buttons at the bottom right.

Figure A.20: Dialog for creating a new relation.

- * Domain selector: with a click on + a new dialog window opens as in A.5.6 for adding concepts. With a click of - all selected concepts are being removed. If a concept was selected when “create relation” action was called then the selected concept is added to the domain selector by the application.
- Range selector: with a click on + a new dialog window opens as in A.5.6 for adding concepts. With a click of - all selected concepts are being removed.
- * Argumentation text field expects an argument why this idea has been introduced. The argument is always classified as “pro”.
- Example check box needs to be ticked if the user provided an example and not an evaluation&justification argument.

Error messages:

- “The given URI is already in use for another concept. Please choose a different URI. Please also consider a different label for the concept.”

A.5.17 Dialog “New attribute”

Type: Dialog window

Name: New attribute (see Fig. A.21)

In this dialog window the user can create a new attribute. The interaction is very similar to the one in A.5.16.

The 'New Attribute' dialog box is a standard Windows-style window with a title bar and a close button. It contains the following elements:

- Label:** A text input field.
- Comment:** A text input field.
- URI:** A combined drop-down menu and text input field.
- Domain:** A large text area with two small buttons (a plus sign and a minus sign) to its right.
- Range:** A drop-down menu with the text 'select data type'.
- Argumentation:** A large text area.
- Example:** A checkbox labeled 'Example'.
- Buttons:** 'ok' and 'cancel' buttons at the bottom right.

Figure A.21: Dialog for creating a new attribute.

Screen elements / Functionality:

- * Label text field expects as input the label of the attribute. If the URI field is empty the entry from the label text field is taken to create a new URI for the attribute.
- Comment text field can be used for additional comments and documentation purposes.
- * URI text field is a combined drop-down-box and text field (comparable to the address bar in a web browser). With the help of the drop-down-box namespaces can be selected for the easy creation of an attribute's URI. To make the URI valid a label needs to be added.
- * Domain selector: with a click on + a new dialog window opens as in A.5.6 for adding concepts. With a click of - all selected concepts are being removed. If a concept was selected when "create relation" action was called then the selected concept is added to the domain selector by the application.
- Range drop-down-box allows the selection of a datatype such as string, integer, number or boolean.
- * Argumentation text field expects an argument why this idea has been introduced. The argument is always classified as "pro".
- Example check box needs to be ticked if the user provided an example and not an evaluation&justification argument.

Error messages:

- “The given URI is already in use for another concept. Please choose a different URI. Please also consider a different label for the concept.”

A.5.18 Dialog “New instance on concept X”

Type: Dialog window

Name: New Instance on Concept X (see Fig. A.22)

The dialog window is titled "New Instance for Concept CONCEPTNAME". It contains the following elements:

- Label:** A text input field.
- Comment:** A text input field.
- URI:** A combined drop-down-box and text input field.
- Table:** A table with two columns: "Relation name" and "Attribute name". The "Relation name" column has a "manage" button next to it. The table has multiple rows for data entry.
- Argumentation:** A large text area for providing additional information.
- Example:** A checkbox labeled "Example".
- Buttons:** "ok" and "cancel" buttons at the bottom right.

Figure A.22: Dialog for creating a new instance.

In this dialog window the user can create a new instance for a certain concept. The problematic aspect of this window is that it is not clear at compile time how many text fields are needed to enter all values for attributes and relations.

Screen elements / Functionality:

- * Label text field expects as input the label of the instance. If the URI field is empty the entry from the label text field is taken to create a new URI for the instance.
- Comment text field can be used for additional comments and documentation purposes.
- * URI text field is a combined drop-down-box and text field (comparable to the address bar in a web browser).. With the help of the drop-down-box namespaces can be selected for the easy creation of an instance’s URI. To make the URI valid a label needs to be added.

- Data entry table: In the data the user enters the data for the attributes or relations. All relation cells have a “manage” button, which opens a new dialog window for selecting instances as in A.5.7 (below). The attribute cells expect a valid data entry. If the entry for a cell is required a * appears in front of the cell.
- * Argumentation text field expects an argument why this idea has been introduced. The argument is always classified as “pro”.
- Example check box needs to be ticked if the user provided an example and not an evaluation&justification argument.

Error messages:

- “The given URI is already in use for another concept. Please choose a different URI. Please also consider a different label for the concept.”
- “The entry in text field LABEL OF TEXTFIELD is not valid because it is not a NUMBER | STRING | INTEGER | BOOLEAN. Please enter a correct value.” This message should be split into at least four different messages.

Reminder messages:

- If some required data entries are not entered by the user this message will appear: “Not all attributes and relations, which require a data entry have a value. The instance will be marked as incomplete. Please come back later to enter the missing data.”

A.5.19 Dialog “Welcome”

Type: Dialog window

Name: Welcome (see Fig. A.23)

This window is the welcome window for the application. If the user starts the application this window is shown. Exception: the user opens up a file and launches the application with a double click on the file.

Screen elements / Functionality:

- Button “Start new ontology development project” opens up a wizard to collect all necessary information for creating a new ontology development project.
- Button “Open ontology development project” opens a dialog for searching a project file to open an existing project. This dialog can also open a configuration file for configuring the application to connect to a server.

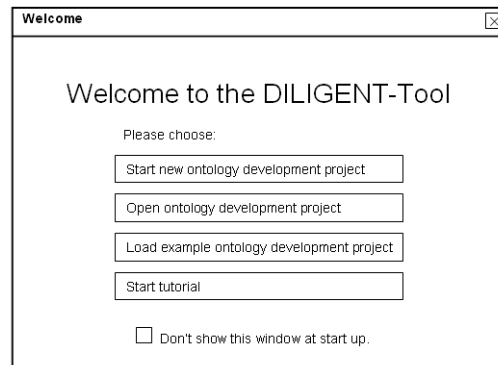


Figure A.23: Welcome dialog.

- Button “Load example ontology development project” opens a project with an already existing ontology that is annotated with an argumentation. It is thought of as a sandbox for exploring with the application.
- Button “Start tutorial” starts a tutorial as briefly described in scenario “Learning the application”.

A.5.20 Dialog “Import text sources”

Type: Dialog window

Name: Import text sources (see Fig. A.24)

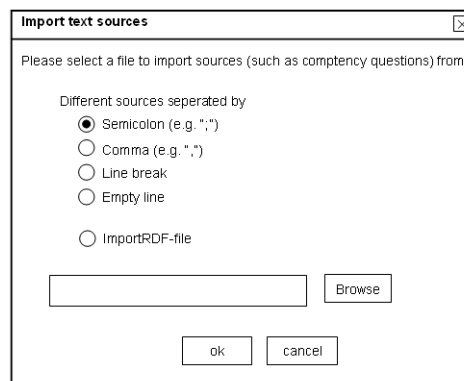


Figure A.24: Dialog for importing text sources.

With this window a number sources for the ontology development in one text file can be imported. This should make the task of creating a new sources easy to handle even if there is a large number of sources.

Screen elements / Functionality:

- Separator radio buttons allow the user to select how different source are separated in the text file.
- Browse text field and button allows selecting a file. A click on browse opens a dialog for browsing to a file.

A.5.21 Dialog “Welcome to project X”

Type: Dialog window

Name: Welcome to project X (see Fig. A.25)

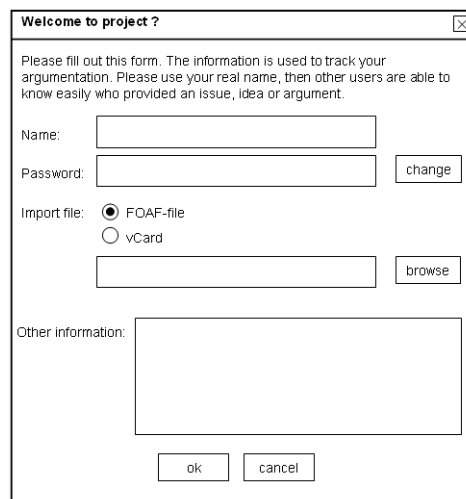


Figure A.25: Dialog for opening a project the first time.

This dialog is shown when the user first logs on to a ontology engineering project. She needs to supply information such as her name and the password that was send via e-mail. The name entered here is used to create

Screen elements / Functionality:

- * Name text field expects the real name of a person. This entry will be used to tag arguments of that user.
- * Password field expects the password of the user that has been sent out via e-mail.
- Button change opens a new dialog window for allowing the user to change her password.
- Other information text field allows the user to enter additional information about himself (e.g. the affiliation she is working, address, contact details, etc.)

- Import FOAF-file or vCard allows the user to import his contact detail using one of these file formats.

A.5.22 Dialog “New idea set”

Type: Dialog window

Name: New idea set (see Fig. A.26)



Figure A.26: Dialog for creating a new idea set.

This dialog window creates a new “idea set”. An idea set is a collection of ideas that create conflicts if it would be introduced into the original ideas. If no idea set has been created all ideas are just listed in the idea view. The user will want to create a new idea set whenever he has an idea, which is in conflict with another existing idea. E.g. the user could suggest a different construction of the concept hierarchy omitting and adding concepts at different places. This is difficult to visualize within the original concept hierarchy. Therefore the concept of idea sets is introduced. It is comparable to a fork in software development. In the end only one idea set will be integrated into the final ontology.

For handling idea sets in a more semantic web like fashion they might internally get an URI assigned, but this is of no concern for the user.

Appendix B

EvA glossary

Argument An argument supports or disagrees with either an issue, elaboration or an idea. It is the way a user can express opinions about modelling the ontology.

Domain expert is a person who is knowledgeable in the domain in which the ontology is being developed. He takes part in the process to ensure that the ontology reflects the domain in which it is modelled.

Elaboration could also be named as sub-issue. It is an issue that is connected to a parent issue and elaborates on that issue.

FOAF-file Friend of a Friend file, a standard for exchanging contact information.

Idea is a possible ontology change operations, but not the actual ontology change operation itself. An ontology change operation will only occur when an idea has been approved either by voting or by the ontology owner. The creation of a modelling primitive is equal to expressing an idea.

Idea set is a group of ideas that relate to one issue. Idea sets are introduced to group modelling primitives (equal to ideas). Idea set will only occur in the application if the discussion comes to a point where it is necessary to introduce contradictory ways of modelling the ontology. A user would create an idea set if he has suggestions which cannot be added to the modelling primitives already present in the editor. Initially there is no idea set present. If a new idea set is created all ideas that have been modelled so far are grouped into the “original idea set” and the rest is grouped into the idea set with the new name.

Issue An issue is a high-level problem that needs to be solved in order to create a useful ontology. It states that some ontology engineering has to be done (e.g. ideas have to be introduced in form of manipulating modelling primitives) to resolve the issue.

Ontology change operation An ontology change operation will occur when one idea or one idea set has been approved either by voting or by the ontology owner. The status of the related issues and ideas is set to agreed.

Ontology engineer is a team member, which is responsible for creating a logically sound ontology. He also needs to take into account the limitations of the modelling language which is used.

Ontology owner is the person who is organizationally responsible for creating and maintaining the ontology. He has the last word in the case of a conflict.

Source can be anything like a competency question, document, URI that is important to look at during the development of the ontology.

System administrator is responsible for maintaining the infrastructure necessary to run the application. He has also some minor tasks in maintaining the application and setting up new development projects.

Task Tasks are used by users to manage the ontology creation. They have a practical management function. More than one task can be connected to one issue. Tasks can also be connected to more than one issue. Tasks can be created manually or after importing all sources and triggering a function that will create tasks accordingly to each source, stating that a user should look at this source.

vCard is a standard for exchanging contact data.

Bibliography

- [App99] D. Appelt. An Introduction to Information Extraction. *Artificial Intelligence Communications*, 12(3):161–172, 1999.
- [ATBC05] N. Aswani, V. Tablan, K. Bontcheva, and H. Cunningham. Indexing and Querying Linguistic Metadata and Document Content. In *Proceedings of Fifth International Conference on Recent Advances in Natural Language Processing (RANLP2005)*, Borovets, Bulgaria, 2005.
- [BCT⁺02] K. Bontcheva, H. Cunningham, V. Tablan, D. Maynard, and O. Hamza. Using GATE as an Environment for Teaching NLP. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies in Teaching NLP*, 2002. <http://gate.ac.uk/sale/acl02/gate4teaching.pdf>.
- [BG04] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004, 2004. available at <http://www.w3.org/TR/rdf-schema/>.
- [BGM05a] J. Brank, M. Grobelnik, and D. Mladenić. Ontology evaluation. SEKT deliverable 1.6.1, Jožef Stefan Institute, 2005.
- [BGM05b] J. Brank, M. Grobelnik, and D. Mladenić. A survey of ontology evaluation techniques. In *Proceedings of the 8th International multi-conference Information Society IS-2005*, Ljubljana, 2005.
- [BMCS02] K. Bontcheva, D. Maynard, H. Cunningham, and H. Saggion. Using Human Language Technology for Automatic Annotation and Indexing of Digital Library Content. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'2002)*, Rome, Italy, 2002. <http://gate.ac.uk/sale/ecdl02/ecdl.pdf>.
- [BR96] G. Booch and J. Rumbaugh. Unified method for object-oriented development. Technical report, Rational Software Corporation, <http://www.relational.com>, 1996.

- [BTMC04] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373, 2004.
- [CBK⁺05] Núria Casellas, Mercedes Blázquez, Atanas Kiryakov, Pompeu Casanovas, and Richard Benjamins. OPJK into PROTON: legal domain ontology integration into an upper-level ontology. In R. Meersman et al., editor, *OTM Workshops 2005, LNCS 3762*, pages 846–855. Springer-Verlag Berlin Heidelberg, 2005.
- [CCP⁺05] Pompeu Casanovas, Núria Casellas, Marta Poblet, J.-J. Vallbé, York Sure, and Denny Vrandečić. Iuriservice ii ontology development. In Pompeu Casanovas, Pablo Noriega, Daniele Bourcier, and V.R.Benjamins, editors, *XXII World Congress of Philosophy of Law and Social Philosophy - Workshop on Artificial Intelligence and Law: The regulation of electronic social systems. Law and the Semantic Web*, number B4 in Special Workshop, pages 327–328, Granada, Spain, MAY 2005. International Association for Philosophy of Law and Social Philosophy, University of Granada. Artificial Intelligence and Law.
- [CCV⁺05] P. Casanovas, N. Casellas, J.J. Vallbé, M. Poblet, F. Ramos, J. Gorroñogointia, J. Contreras, M. Blázquez, and R. Benjamins. Iuriservice ii: Ontology development and architectural design. In *Proceedings of the Internacional Conference on Artificial Intelligence and Law, ICAIL-05, AAI-ACM*, pages 188–195, Bologna, Italy, 2005.
- [CL96] J. Cowie and W. Lehnert. Information Extraction. *Communications of the ACM*, 39(1):80–91, 1996.
- [CL01] Ward Cunningham and Bo Leuf. *The Wiki Way. Quick Collaboration on the Web*. Addison-Wesley, 2001.
- [CMBT02] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.
- [CPC⁺04] Pompeu Casanovas, Marta Poblet, Núria Casellas, Joan-Josep Vallbé, Francisco Ramos, Richard Benjamins, Mercedes Blázquez, Luis Rodrigo, Jesús Contreras, and Jesús Gorroñogointia Cruz. Legal scenario: Case study-intelligent integrated decision support for legal professionals. SEKT deliverable 10.2.1, Intelligent Software Components S.A. and Universitat Autònoma de Barcelona, 2004.

- [CPC⁺05] P. Casanovas, M. Poblet, N. Casellas, J. Contreras, R. Benjamins, and M. Blázquez. Supporting newly-appointed judges: a legal knowledge management case study. *Journal of Knowledge Management*, 9(5):7–27, 2005.
- [CR03] Alan Cooper and Robert M. Reimann. *About Face 2.0: The Essentials of Interaction Design*. Wiley and Sons, 2003.
- [CT02] Nigel Collier and Koichi Takeuchi. PIA-Core: Semantic Annotation through Example-based Learning. In *Proceedings of the LREC 2002 Third International Conference on Language Resources and Evaluation*, pages 1611–1614, Las Palmas, Canary Islands - Spain, 2002.
- [Cun99] H. Cunningham. Information Extraction: a User Guide (revised version). Research Memorandum CS-99-07, Department of Computer Science, University of Sheffield, May 1999.
- [dMA03] A. de Moor and M. Aakhus. Argumentation support: From technologies to tools. In *Proc. of the 8th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2003)*, Tilburg, The Netherlands, June 1-2 2003.
- [EHLPO4] Nick Edgar, Kevin Haaland, Jin Li, and Kimberley Peter. Eclipse user interface guidelines version 2.1. Technical report, IBM, February 2004.
- [FSW01] Simon Fong, Aixin Sun, and Kin Keong Wong. Price Watcher Agent for E-Commerce. In *Proceedings of the second Asia-Pacific Conference on Intelligent Agent Technology (IAT-2001)*, pages 294–299, Maebashi Terrsa, Maebashi City, Japan, 2001.
- [FW04] David C. Fallside and Priscilla Walmsley. Xml schema part 0: Primer second edition, Oct 2004.
- [GGM⁺02] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In A. Gómez-Pérez and V. R. Benjamins, editors, *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (EKAW 2002)*, volume 2473 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 166–181, Siguenza, Spain, 2002. Springer.
- [GH06] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. *Journal of Information Science*, 2006. to appear.
- [GM06] M. Grobelnik and D. Mladenić. Knowledge discovery for ontology construction. In J. Davies, R. Studer, and P. Warren, editors, *Semantic Web: trends and research*, chapter 2. Wiley, 2006. to appear.

- [GMG05a] M. Grčar, D. Mladenić, and M. Grobelnik. Applying collaborative filtering to real-life corporate data. In *Proceedings of the 29th Annual Conference of the German Classification Society (GfKI 2005)*. Springer, 2005.
- [GMG05b] M. Grčar, D. Mladenić, and M. Grobelnik. Data quality issues in collaborative filtering. In *Proceedings of ESWC-2005 Workshop on End User Aspects of the Semantic Web*, Heraklion, Greece, May 2005.
- [GPFLC03] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer, 2003.
- [GPS98] A. Gangemi, D.M. Pisanelli, and G. Steve. Ontology integration: Experiences with medical terminologies. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 163–178, Amsterdam, 1998. IOS Press.
- [Gru95] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928, 1995.
- [GSV04] T. Gabel, Y. Sure, and J. Voelker. KAON – ontology management infrastructure. SEKT informal deliverable 3.1.1.a, Institute AIFB, University of Karlsruhe, 2004.
- [GW98] R. Gaizauskas and Y. Wilks. Information Extraction: Beyond Document Retrieval. *Journal of Documentation*, 54(1):70–105, 1998.
- [HV04] P. Haase and J. Voelker. Requirements analysis for usage-driven and data-driven change discovery. SEKT informal deliverable 3.3.1.a, Institute AIFB, University of Karlsruhe, 2004.
- [JM05] A. Jakulin and D. Mladenić. Ontology grounding. In *Proceedings of the 8th International multi-conference Information Society IS-2005*, pages 170–173, 2005.
- [Joa98] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, USA, 1998.
- [KF01] M. Klein and D. Fensel. Ontology versioning for the Semantic Web. In *Proc. of the First Int. Semantic Web Working Symposium (SWWS)*, pages 75–91, 2001.
- [KOM05] A. Kiryakov, D. Ognyanov, and D. Manov. Owlrim a pragmatic semantic repository for owl. In *Proc. of International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005) at WISE 2005*, New York City, USA, November 2005.

- [KVV05] Markus Krtzsch, Denny Vrandečić, and Max Vlk. Wikipedia and the Semantic Web – the missing links. In *Proceedings of the 1st International Wikimedia Conference, Wikimania 2005*, Aug 2005.
- [LMG05] L. Lancieri, D. Maynard, and F. Gandon. Success stories and best practices. Technical Report D1.4.2, KnowledgeWeb Deliverable, 2005.
- [LR02] P. Lal and S. Ruger. Extract-based summarization with simplification. In *Proceedings of the ACL 2002 Automatic Summarization / DUC 2002 Workshop*, 2002.
- [Mat04] Adam Mathes. Folksonomies – cooperative classification and communication through shared metadata. *Computer Mediated Communication*, Dec 2004.
- [MBC04] D. Maynard, K. Bontcheva, and H. Cunningham. Automatic Language-Independent Induction of Gazetteer Lists. In *Proceedings of 4th Language Resources and Evaluation Conference (LREC'04)*, 2004. <http://gate.ac.uk/sale/lrec2004/gazcollector.pdf>.
- [MBS⁺02] D. Maynard, K. Bontcheva, H. Saggion, H. Cunningham, and O. Hamza. Using a Text Engineering Framework to Build an Extendable and Portable IE-based Summarisation System. In *Proceedings of the ACL Workshop on Text Summarisation*, 2002. <http://gate.ac.uk/sale/hsl/hsl.pdf>.
- [MM04] F. Manola and E. Miller. Resource Description Framework (RDF). primer. W3C Recommendation 10 February 2004, 2004. available at <http://www.w3.org/TR/rdf-primer/>.
- [MSS⁺03] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. *SEmantic portAL – The SEAL approach*, chapter 11, pages 461–518. MIT Press, 2003.
- [MT87] W. C. Mann and S. A. Thompson. Rhetorical structure theory: A theory of text organization. In L. Polanyi, editor, *The Structure of Discourse*. Ablex Publishing Corporation, Norwood, N.J., 1987.
- [MTBC03] D. Maynard, V. Tablan, K. Bontcheva, and H. Cunningham. Rapid customisation of an Information Extraction system for surprise languages. *Special issue of ACM Transactions on Asian Language Information Processing: Rapid Development of Language Capabilities: The Surprise Languages*, 2003.
- [Noy02] N. Noy. The OntoWeb evaluation experiment for ontology editors: Using Protégé-2000 to represent the travel domain. In Sure and Angele [SA02], pages 103–107. CEUR-WS Publication, available at <http://CEUR-WS.org/Vol-62/>.

- [PB88] C. Potts and G. Bruns. Recording the reasons for design decisions. In *Proceedings of the 10th international conference on Software engineering*, pages 418–427. IEEE Computer Society Press, 1988.
- [PBC⁺05] Raúl Peña, Mercedes Blázquez, Jesús Contreras Cino, Richard Benjamins, Pompeu Casanovas, and Núria Casellas. Legal case study prototype informal deliverable. SEKT Informal Deliverable 10.3.1.1, Intelligent Software Components S.A. and Universitat Autònoma de Barcelona, 2005.
- [PC05] M. Poblet and P. Casanovas. *Recruitment, Professional Evaluation and Career of Judges and Prosecutors in Spain*, pages 185–213. IRSIG-CNR, University of Bologna, Ed. Lo Scarabeo, 2005.
- [PM01] H. S. Pinto and J. P. Martins. A methodology for ontology integration. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP2001)*, pages 131–138, New York, 2001. ACM Press.
- [PSST04] H. S. Pinto, S. Staab, Y. Sure, and C. Tempich. OntoEdit empowering SWAP: a case study in supporting Distributed, Loosely-controlled and evolving Engineering of ontologies (DILIGENT). In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, *First European Semantic Web Symposium, ESWS 2004*, volume 3053 of *LNCIS*, pages 16–30, Heraklion, Crete, Greece, May 2004. Springer.
- [RCCP04] L. Rodrigo, M. Blázquez Cívico, P. Casanovas, and M. Poblet. Legal case study before analysis. SEKT deliverable 10.1.1, Intelligent Software Components S.A. and Universitat Autònoma de Barcelona, 2004.
- [SA02] Y. Sure and J. Angele, editors. *Proceedings of the First International Workshop on Evaluation of Ontology based Tools (EON 2002)*, volume 62 of *CEUR Workshop Proceedings*, Sigüenza, Spain, 2002. CEUR-WS Publication, available at <http://CEUR-WS.org/Vol-62/>.
- [SCB⁺02] H. Saggion, H. Cunningham, K. Bontcheva, D. Maynard, C. Ursu, O. Hamza, and Y. Wilks. Access to Multimedia Information through Multisource and Multilanguage Information Extraction. In *Proceedings of the 7th Workshop on Applications of Natural Language to Information Systems (NLDB 2002)*, Stockholm, Sweden, 2002.
- [SCM⁺02] H. Saggion, H. Cunningham, D. Maynard, K. Bontcheva, O. Hamza, C. Ursu, and Y. Wilks. Extracting Information for Information Indexing of Multimedia Material. In *Proceedings of 3rd Language Resources and Evaluation Conference (LREC'2002)*, 2002. http://gate.ac.uk/sale/lrec2002/mumis_lrec2002.ps.
- [sek] Technical report.

- [SEK04] Language issues - software v1. SEKT deliverable 1.4.1, Jožef Stefan Institute, 2004.
- [SEK05a] Bt digital library prototype. SEKT deliverable 10.3.1, British Telecommunications, 2005.
- [SEK05b] Ontology generation from scratch - software v1.0. SEKT deliverable 1.7.1, Jožef Stefan Institute, 2005.
- [SEK05c] Massive automatic annotation v1. SEKT deliverable 2.6.1, University of Sheffield, 2005.
- [SEK05d] Search and browse facility final version. SEKT deliverable 5.2.2, (British Telecom), 2005.
- [SEK05e] Integration of gate with kaon v1. SEKT deliverable 6.6.4, Institute AIFB, University of Karlsruhe, 2005.
- [SEK05f] Siemens prototype. SEKT formal deliverable 9.3.1, Siemens Business Services, DEC 2005.
- [She00] Colin Shearer. The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, 2000.
- [STV⁺05] Y. Sure, C. Tempich, D. Vrandečić, H.S. Pinto, E. Paslaru Bontas, and M. Hefke. Sekt methodology: Evaluation of guidelines. SEKT formal deliverable 7.1.2, Institute AIFB, University of Karlsruhe, DEC 2005.
- [SWM04] M. K. Smith, C. Welty, and D. McGuinness. OWL Web Ontology Language Guide, 2004. W3C Recommendation 10 February 2004, available at <http://www.w3.org/TR/owl-guide/>.
- [TBMC03] V. Tablan, K. Bontcheva, D. Maynard, and H. Cunningham. OL-LIE: On-Line Learning for Information Extraction. In *Proceedings of the HLT-NAACL Workshop on Software Engineering and Architecture of Language Technology Systems*, Edmonton, Canada, 2003. <http://gate.ac.uk/sale/hlt03/ollie-sealts.pdf>.
- [TKM04] I. Terziev, A. Kiryakov, and D. Manov. Base upper-level ontology (bulo) guidance. SEKT deliverable 1.8.1, Ontotext Lab, Sirma AI EAD (Ltd.), 2004.
- [TPSS05] Christoph Tempich, Sofia Pinto, York Sure, and Steffen Staab. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (diligent). In *Second European Semantic Web Conference, ESWC 2005*, volume 3532 of *LNCS*, pages 241–256, 2005.

- [TSVP04] Christoph Tempich, York Sure, Denny Vrandečić, and H. Sofia Pinto. SEKT methodology: Initial guidelines. SEKT formal deliverable 7.1.1, Institute AIFB, University of Karlsruhe, DEC 2004.
- [TUB⁺02] V. Tablan, C. Ursu, K. Bontcheva, H. Cunningham, D. Maynard, O. Hamza, Tony McEnery, Paul Baker, and Mark Leisher. A Unicode-based Environment for Creation and Use of Language Resources. In *3rd Language Resources and Evaluation Conference*, 2002. <http://gate.ac.uk/sale/iesl03/iesl03.pdf>.
- [UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, Montreal, Canada, 1995.
- [VMF⁺05] Joan-Josep Vallbé, M. Antnia Martí, Blaž Fortuna, Aleks Jakulin, Dunja Mladenič, and Pompeu Casanovas. Stemming and lemmatisation. improving knowledge management through language processing techniques. In *Proceedings of the B4 Workshop on Artificial intelligence and Law. IVR'05*, Granada, Spain, 2005. Available at: <http://www.lefis.org>.
- [Vol04] R. Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, University of Karlsruhe, 2004.
- [VPST05] Denny Vrandečić, H. Sofia Pinto, York Sure, and Christoph Tempich. The diligent knowledge processes. *Journal of Knowledge Management*, 9(5):85–96, Oct 2005.
- [VVS05] J. Völker, D. Vrandečić, and Y. Sure. Automatic evaluation of ontologies (AEON). In *Proc. of the 4th International Semantic Web Conference (ISWC'05)*, Nov 2005.
- [Wal05] Jimbo Wales. Wikipedia and the free culture revolution. OOPSLA/WikiSym Invited Talk, Oct 2005.
- [WLT⁺03] M. M. Wood, S. J. Lydon, V. Tablan, D. Maynard, and H. Cunningham. Using parallel texts to improve recall in IE. In *Recent Advances in Natural Language Processing*, Bulgaria, 2003.